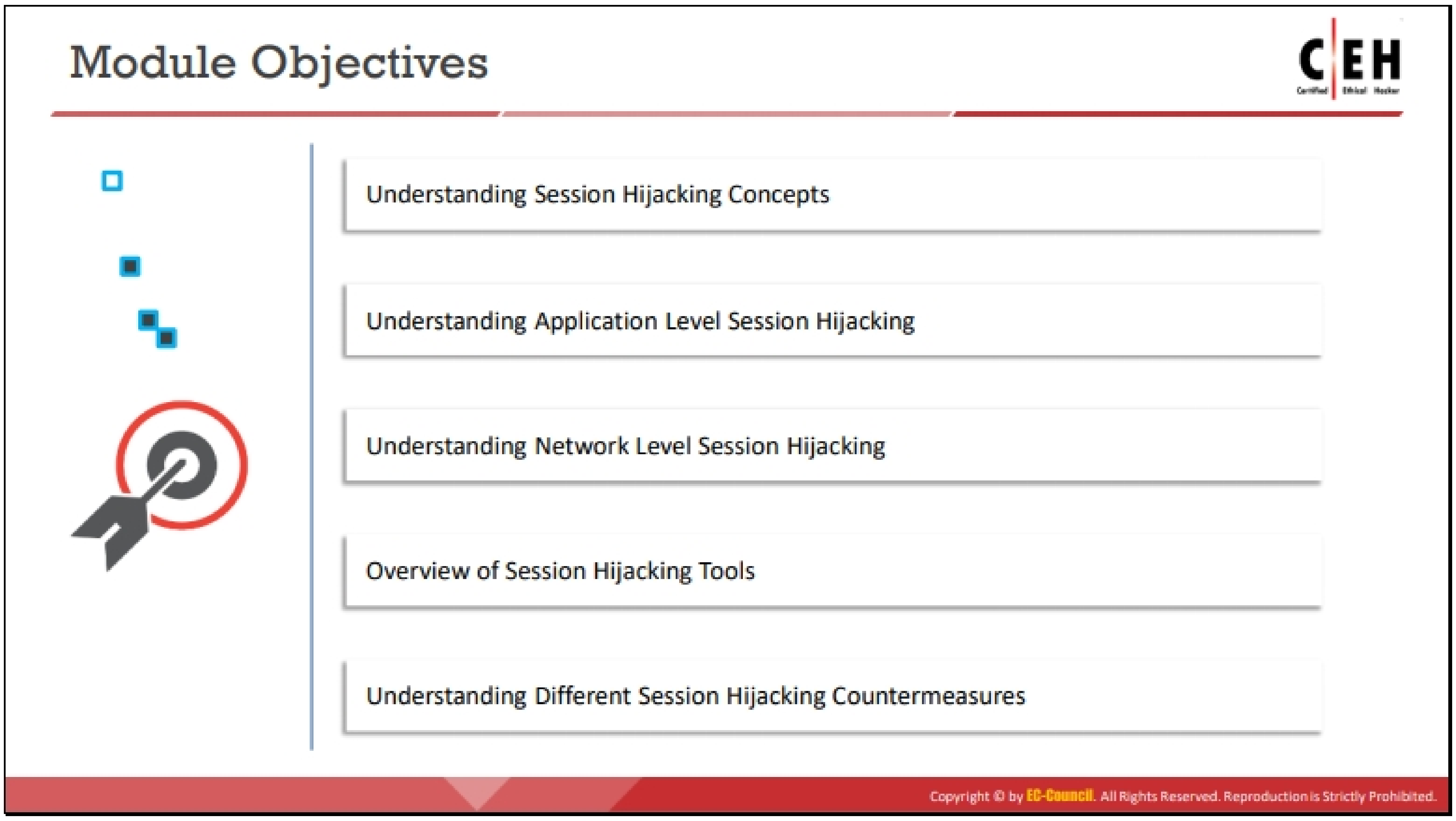
C|EH™

Certified Ethical Hacker

Module 11:
Session Hijacking



Module Objectives

- Understanding Session Hijacking Concepts
- Understanding Application Level Session Hijacking
- Understanding Network Level Session Hijacking
- Overview of Session Hijacking Tools
- Understanding Different Session Hijacking Countermeasures

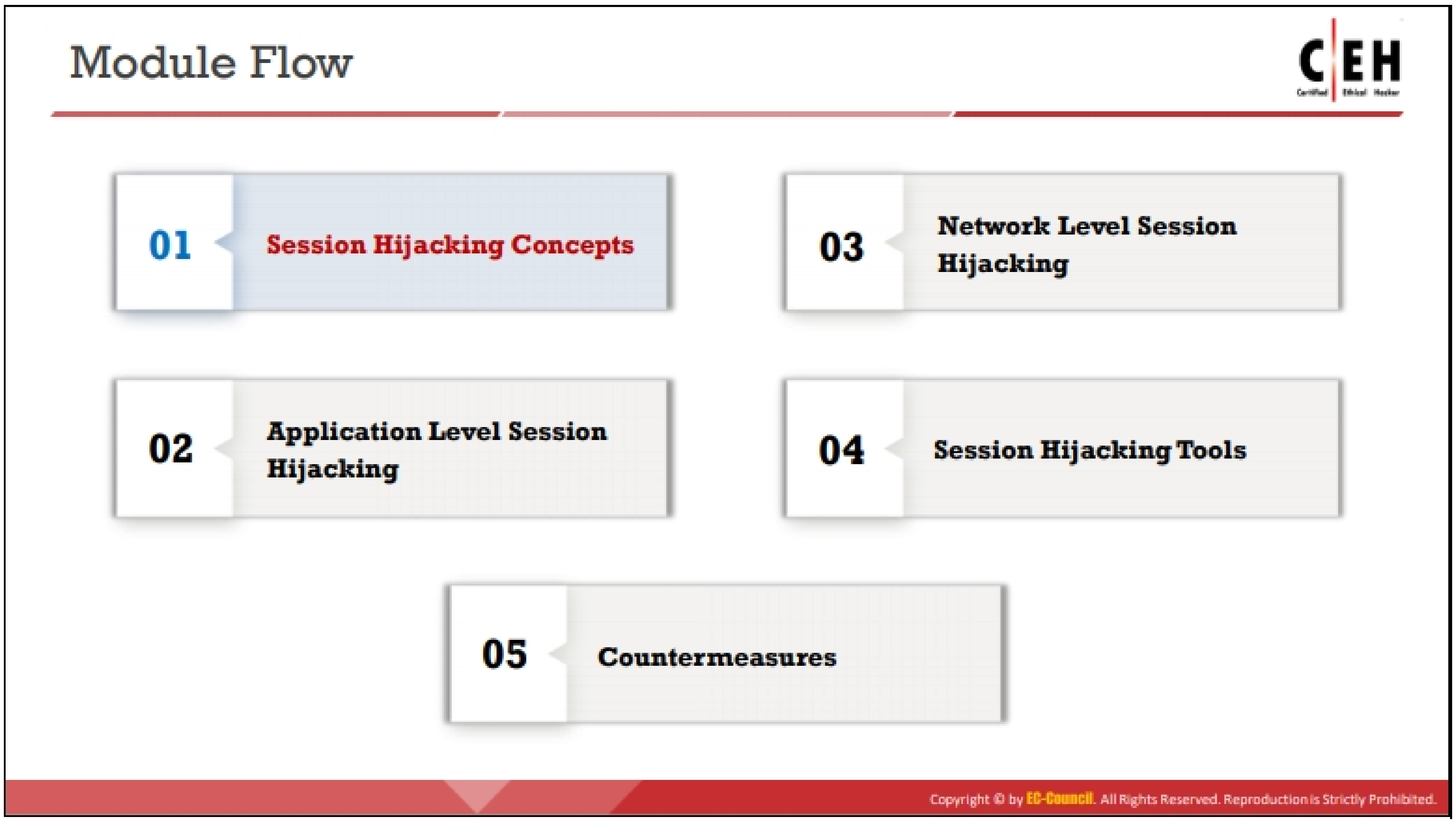
Module Objectives

Session hijacking allows attackers to take over an active session by bypassing the authentication process. Thereafter, they can perform any action on the hijacked system.

This module aims to provide comprehensive information on session hijacking. It starts with an introduction to session hijacking concepts and provides insight into session hijacking at the application and network levels. Later, the module discusses tools used to hijack a session between a client and server. It also discusses various countermeasures to defend against session hijacking attacks.

At the end of this module, you will be able to do the following:

- Describe session hijacking concepts
- Perform application level session hijacking
- Perform network level session hijacking
- Use different session hijacking tools
- Apply session hijacking countermeasures



Module Flow

01 Session Hijacking Concepts

02 Application Level Session Hijacking

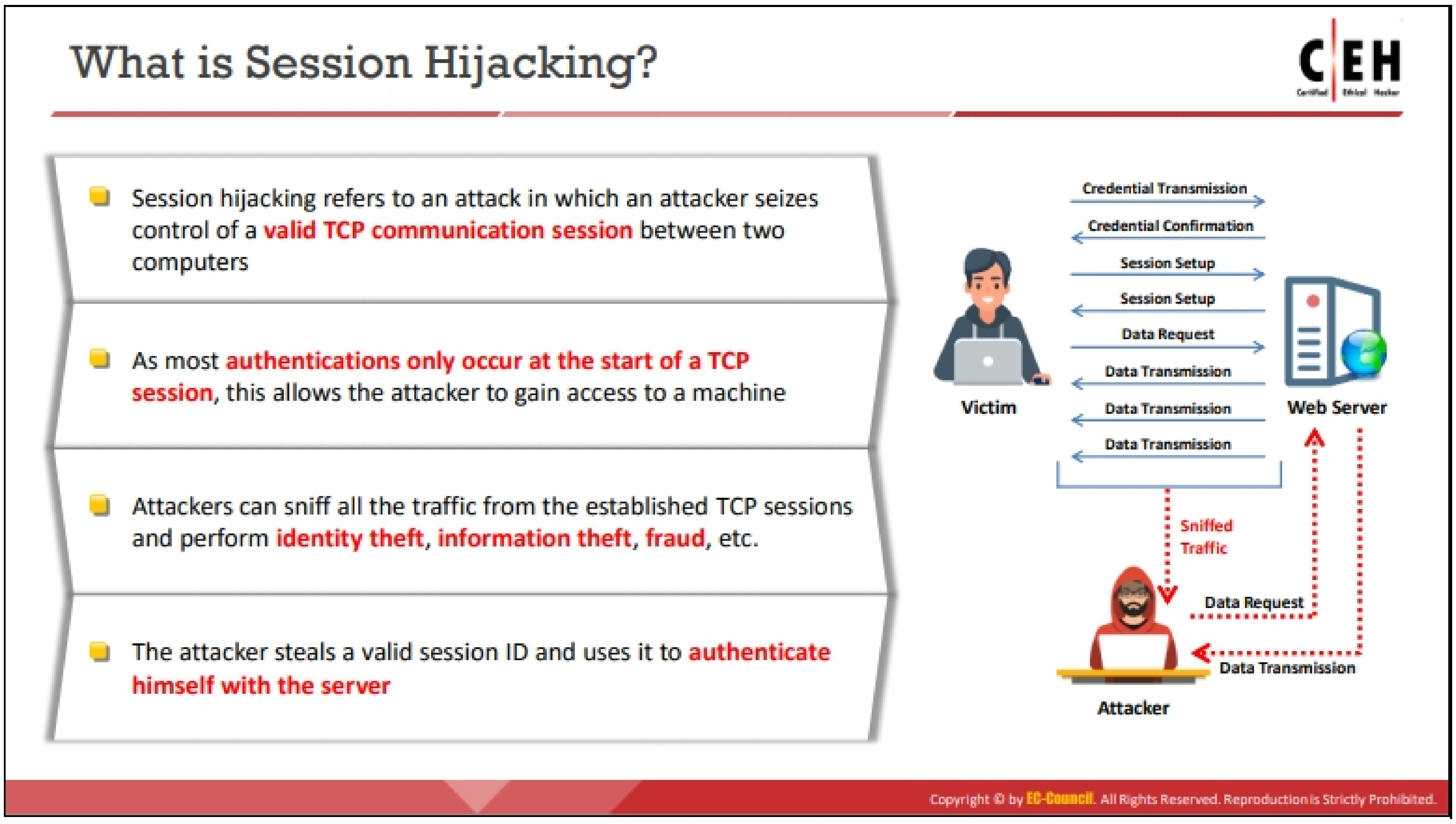
03 Network Level Session Hijacking

04 Session Hijacking Tools

05 Countermeasures

Session Hijacking Concepts

Familiarization with basic concepts related to session hijacking is important to attain a comprehensive understanding. This section explains what session hijacking is as well as the reasons why session hijacking succeeds. It also discusses the session hijacking process, packet analysis of a local session hijack, types of session hijacking, session hijacking in an Open Systems Interconnection (OSI) model, and differences between spoofing and hijacking.



What is Session Hijacking?

- Session hijacking refers to an attack in which an attacker seizes control of a **valid TCP communication session** between two computers
- As most **authentications only occur at the start of a TCP session**, this allows the attacker to gain access to a machine
- Attackers can sniff all the traffic from the established TCP sessions and perform **identity theft**, **information theft**, **fraud**, etc.
- The attacker steals a valid session ID and uses it to **authenticate himself with the server**

What is Session Hijacking?

A web server sends a session identification token or key to a web client after successful authentication. These session tokens differentiate multiple sessions that the server establishes with clients. Web servers use various mechanisms to generate random tokens and controls to secure the tokens during transmission.

Session hijacking is an attack in which an attacker takes over a valid Transmission Control Protocol (TCP) communication session between two computers. Because most types of authentication are performed only at the start of a TCP session, an attacker can gain access to a machine while a session is in progress. Attackers can sniff all the traffic from established TCP sessions and perform identity theft, information theft, fraud, etc.

A session hijacking attack exploits a session-token generation mechanism or token security controls so that the attacker can establish an unauthorized connection with a target server. The attacker can guess or steal a valid session ID, which identifies authenticated users, and use it to establish a session with the server. The web server responds to the attacker's requests under the impression that it is communicating with an authenticated user.

Attackers can use session hijacking to launch various kinds of attacks, such as man-in-the-middle (MITM) and denial-of-service (DoS) attacks. In an MITM attack, an attacker places themselves between an authorized client and a server by performing session hijacking to ensure that information flowing in either direction passes through them. However, the client and server believe they are directly communicating with each other. Attackers can also sniff sensitive information and disrupt sessions to launch a DoS attack.

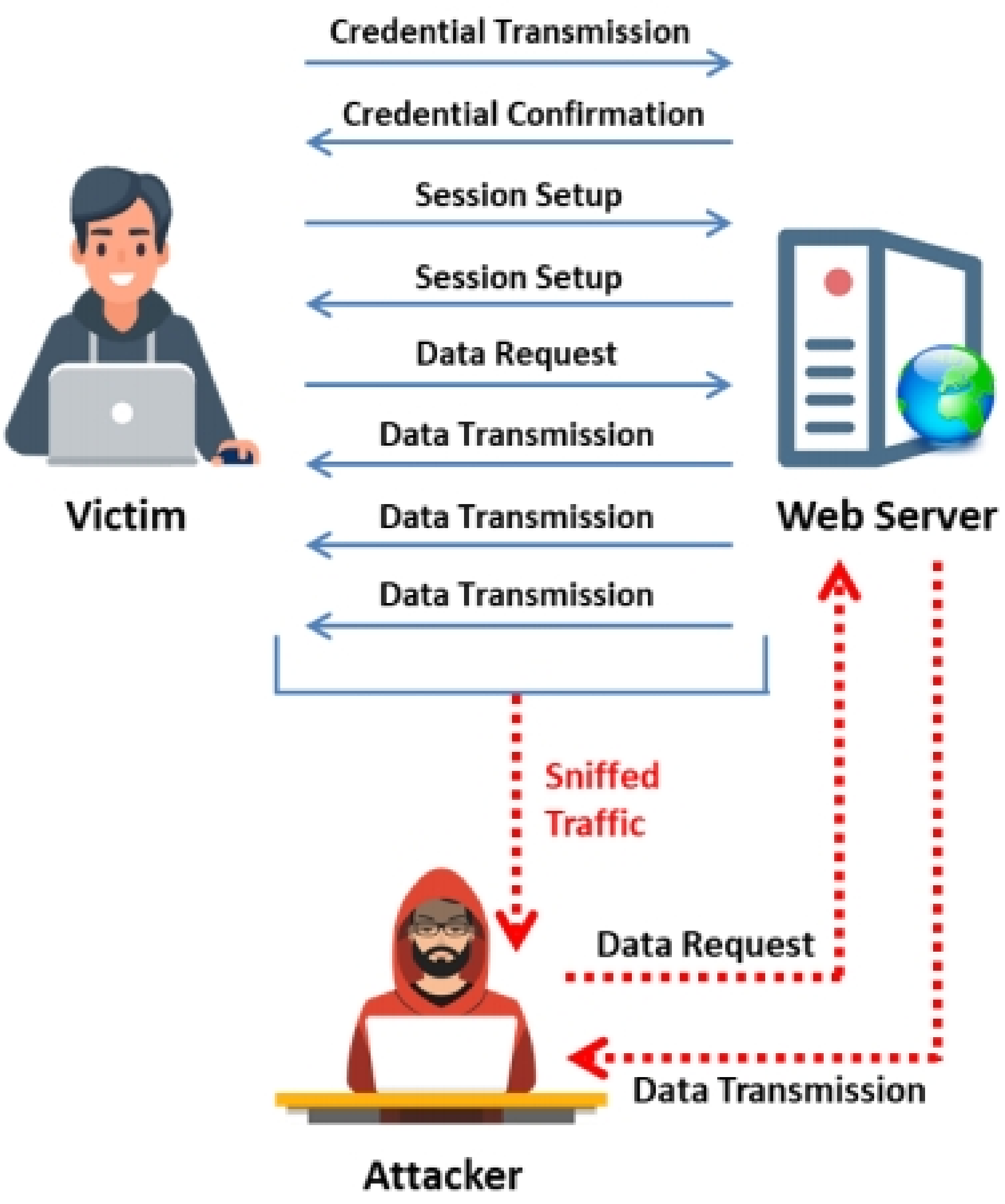
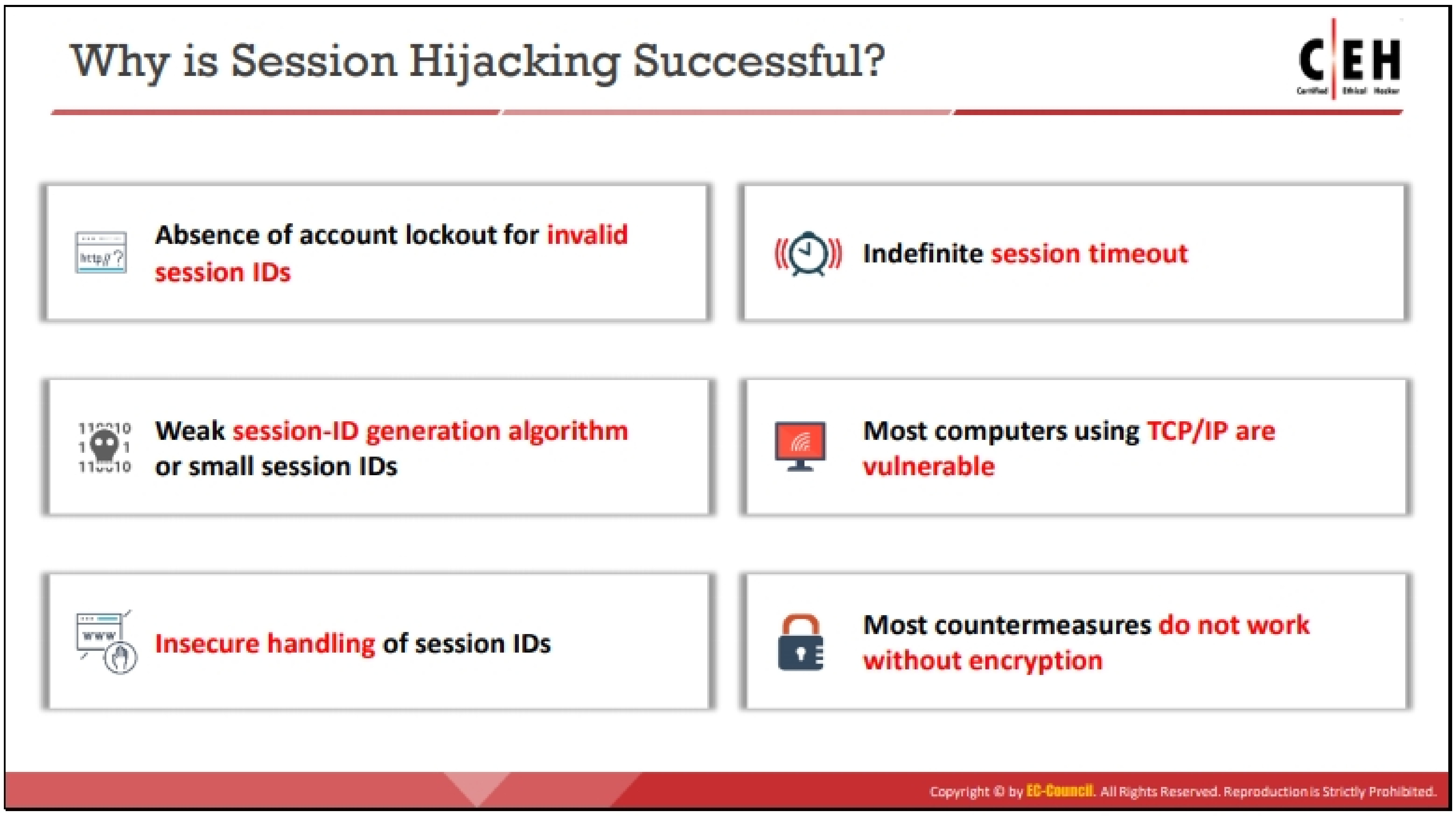

Credential Transmission
Credential Confirmation
Session Setup
Session Setup
Data Request
Data Transmission
Data Transmission
Data Transmission
Victim
Web Server
Sniffed Traffic
Data Request
Data Transmission
Attacker

Figure 11.1: Example of session hijacking



Why is Session Hijacking Successful?

Absence of account lockout for invalid session IDs

Indefinite session timeout

Weak session-ID generation algorithm or small session IDs

Most computers using TCP/IP are vulnerable

Insecure handling of session IDs

Most countermeasures do not work without encryption

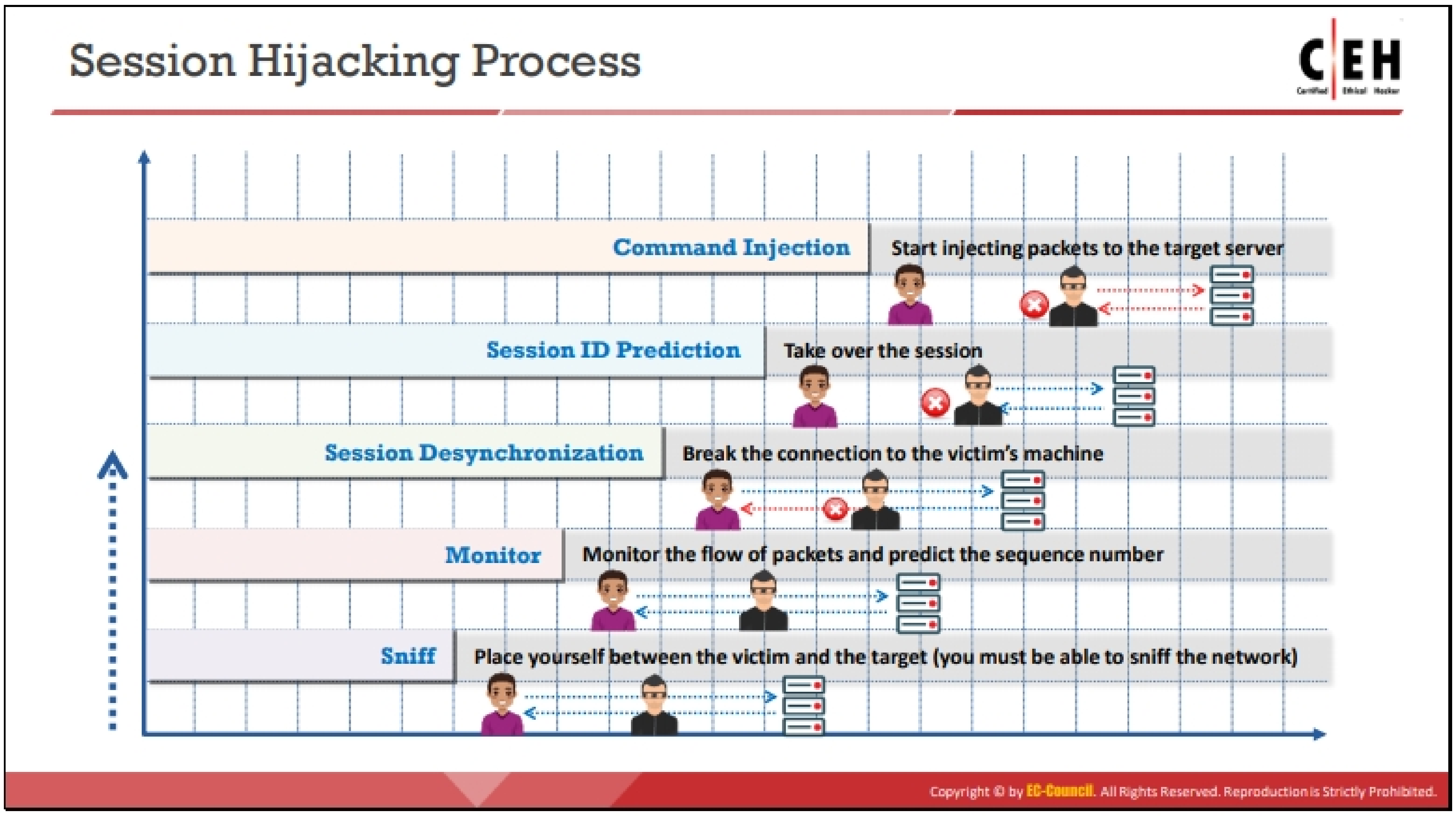
Why is Session Hijacking Successful?

Session hijacking succeeds because of the following factors.

- **Absence of account lockout for invalid session IDs**: If a website does not implement account lockout, an attacker can make several attempts to connect with varying session IDs embedded in a genuine URL. The attacker can continue making attempts until the actual session ID is determined. This attack is also known as a brute-force attack. During a brute-force attack, the web server does not display a warning message or complaint, allowing the attacker to determine the valid session ID.
- **Weak session-ID generation algorithm or small session IDs**: Most websites use linear algorithms to predict variables such as time or IP address for generating session IDs. By studying the sequential pattern and generating multiple requests, an attacker can easily narrow the search space necessary to forge a valid session ID. Even if a strong session-ID generation algorithm is used, an active session ID can be easily determined if the string is short.
- **Insecure handling of session IDs**: An attacker can retrieve stored session-ID information by misleading the user's browser into visiting another site. Before the session expires, the attacker can exploit the information in many ways, such as Domain Name System (DNS) poisoning, cross-site scripting exploitation, and the exploitation of a bug in the browser.
- **Indefinite session timeout**: Session IDs with an indefinite expiration time provides an attacker with unlimited time to guess a valid session ID. An example of this is the "remember me" option in many websites. The attacker can use static session IDs to the user's web account after capturing the user's cookie file. The attacker can also perform

session hijacking if they can break into a proxy server, which potentially logs or caches session IDs.

- **Most computers using TCP/Internet Protocol (IP) are vulnerable**: All machines running TCP/IP are vulnerable to session hijacking because of the design flaws inherent in TCP/IP.
- **Most countermeasures do not work without encryption**: It is easy to sniff session IDs in a flat network if transport security is not set up properly during the transmission of session ID cookies, even if a web application uses Secure Sockets Layer (SSL) encryption. An attacker's task becomes even easier if they capture session IDs containing actual login information.



Session Hijacking Process

Command Injection Start injecting packets to the target server

Session ID Prediction Take over the session

Session Desynchronization Break the connection to the victim's machine

Monitor Monitor the flow of packets and predict the sequence number

Sniff Place yourself between the victim and the target (you must be able to sniff the network)

Session Hijacking Process

It is easier for an attacker to sneak into a system as a genuine user than to enter a system directly. An attacker can hijack a genuine user's session by finding an established session and taking it over after user authentication. After hijacking the session, the attacker can stay connected for hours without arousing suspicion. During this period, all traffic intended for the user's IP address goes to the attacker's system instead, and the attacker can plant backdoors or gain additional access to the system. Here, we examine how an attacker hijacks a session.

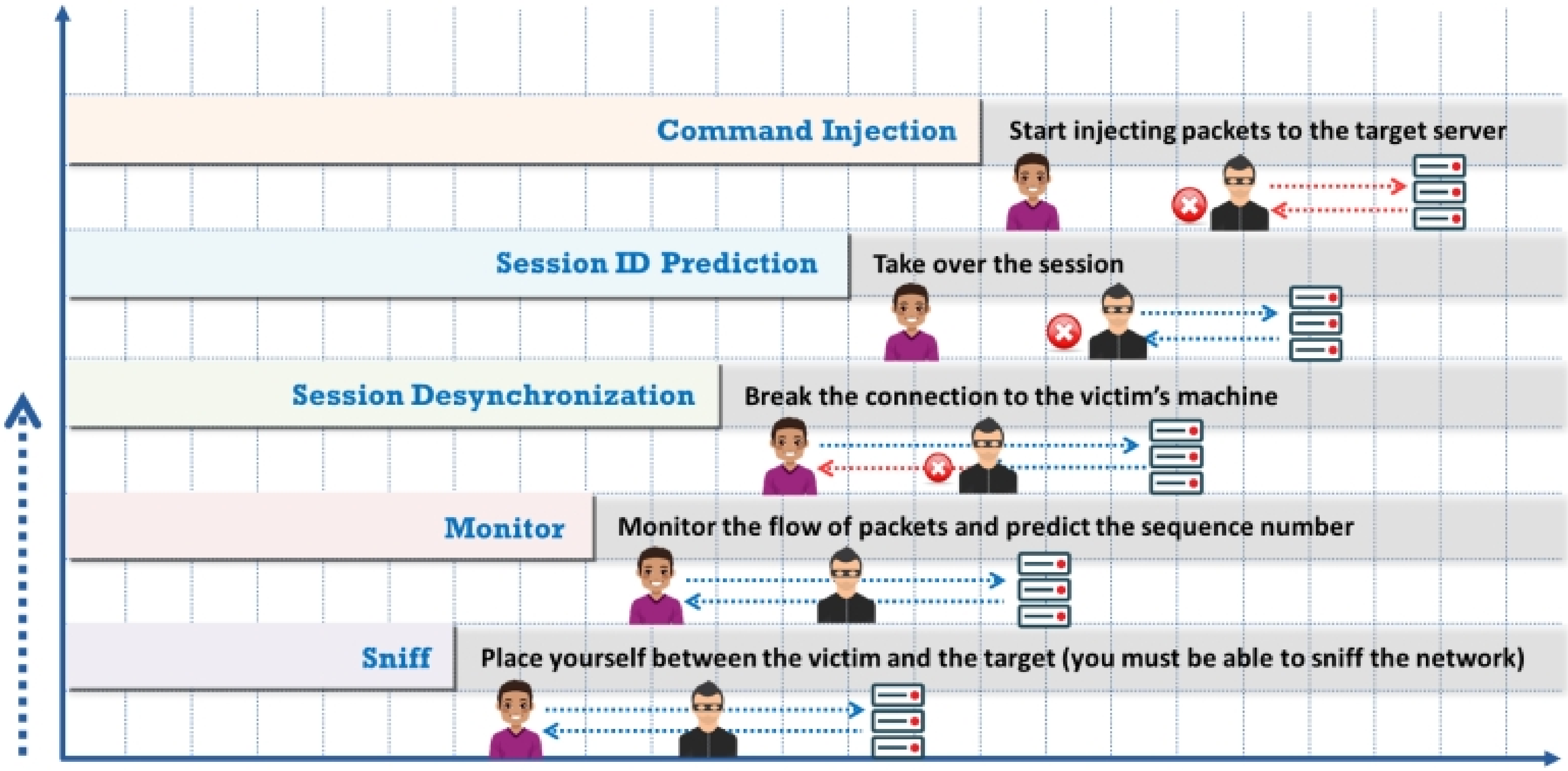

Command Injection
Start injecting packets to the target server
Session ID Prediction
Take over the session
Session Desynchronization
Break the connection to the victim's machine
Monitor
Monitor the flow of packets and predict the sequence number
Sniff
Place yourself between the victim and the target (you must be able to sniff the network)

Figure 11.2: Session hijacking process

Session hijacking can be divided into three broad phases.

- **Tracking the connection**

  The attacker uses a network sniffer to track a victim and host or uses a tool such as Nmap to scan the network for a target with a TCP sequence that is easy to predict. After identifying a victim, the attacker captures the sequence and acknowledgment numbers of the victim because TCP checks these numbers. The attacker then uses these numbers to construct packets.

- **Desynchronizing the connection**

  A desynchronized state occurs when a connection between a target and host is established, or stable with no data transmission or the server's sequence number is not equal to the client's acknowledgment number, or vice versa.

  To desynchronize the connection between the target and host, the attacker must change the sequence number or acknowledgment number (SEQ/ACK) of the server. For this purpose, the attacker sends null data to the server; consequently, the server's SEQ/ACK numbers advance, while the target machine does not register the increment. For example, before desynchronization, the attacker monitors the session without any interference, following which they send a large amount of null data to the server. These data change the ACK number on the server without affecting anything else, thereby desynchronizing the server and target.

  Another approach is to send a reset flag to the server to break the connection on the server side. Ideally, this occurs in the early setup stage of the connection. The attacker's goal is to break the connection on the server side and create a new connection with a different sequence number.

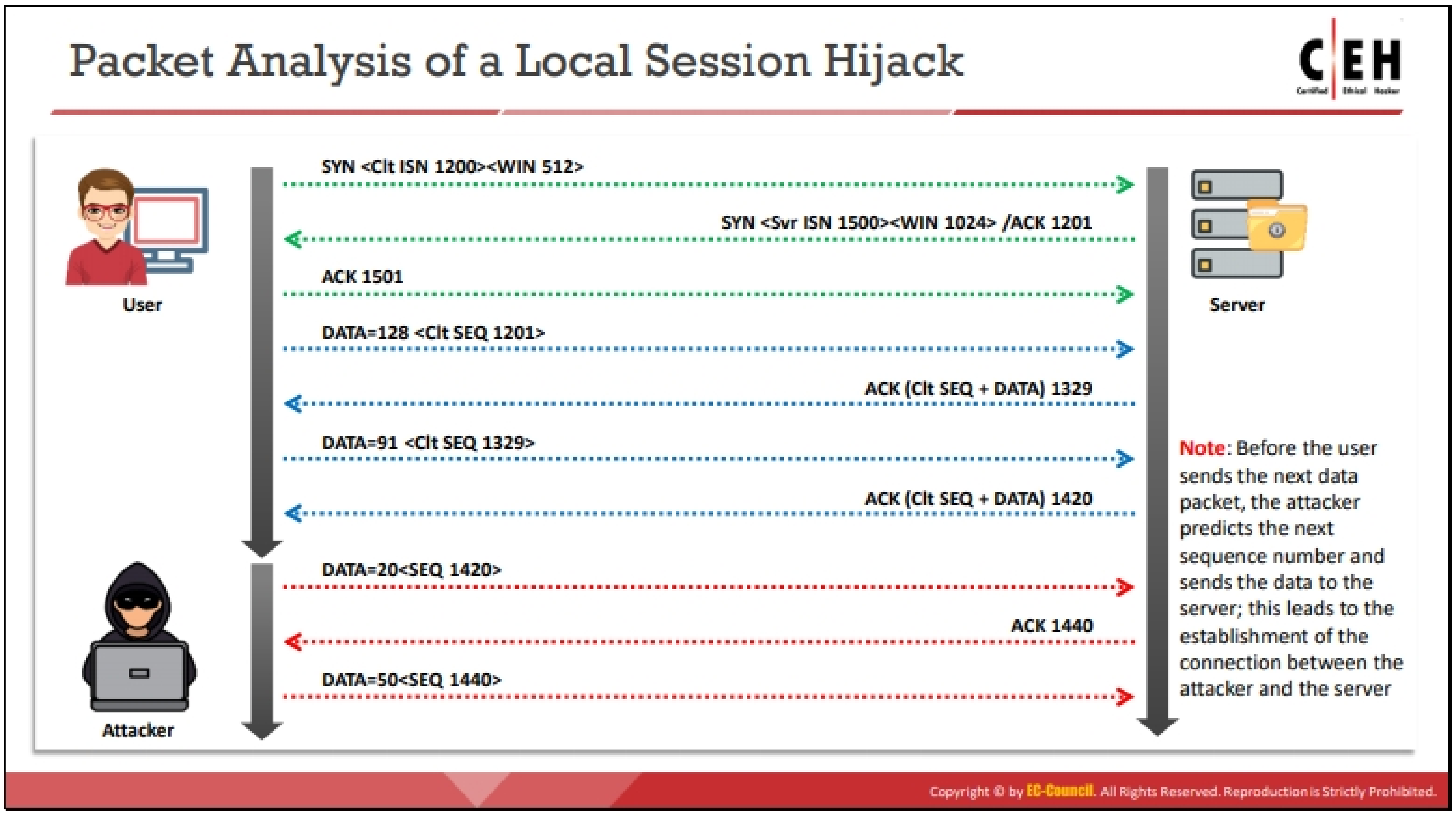
  The attacker waits for a SYN/ACK packet from the server to the host. On detecting a packet, the attacker immediately sends an RST packet and a SYN packet with identical parameters, such as a port number with a different sequence number, to the server. The server, on receiving the RST packet, closes the connection with the target and initiates another one based on the SYN packet but with a different sequence number on the same port. After opening a new connection, the server sends a SYN/ACK packet to the target for acknowledgement. The attacker detects (but does not intercept) this packet and sends an ACK packet to the server. Now, the server is in the established state. The aim is to keep the target conversant and ensure that it switches to the established state on receiving the first SYN/ACK packet from the server. Consequently, both the server and target are desynchronized but in an established state.

  An attacker can also use a FIN flag, but this will make the server respond with an ACK packet, thus revealing the attack through an ACK storm. The attack is revealed because of a flaw in this method of hijacking a TCP connection. While receiving an unacceptable packet, the host acknowledges it by sending the expected sequence number. This unacceptable packet generates an ACK packet, thereby creating an endless loop for every data packet. The mismatch in SEQ/ACK numbers results in excess network traffic with both the server and target attempting to verify the correct sequence. Because these packets carry no data, retransmission does not occur if the packet is lost. However, because TCP uses IP, the loss of a single packet ends the unwanted conversation between the server and target.

An attacker can add a desynchronizing stage to the hijack sequence to deceive the target host. Without desynchronizing, the attacker injects data into the server while keeping their identity hidden by spoofing an IP address. However, the attacker should ensure that the server responds to the target host as well.

- **Injecting the attacker's packet**

  Once the attacker has interrupted the connection between the server and target, they can either inject data into the network or actively participate as the man in the middle, passing data from the target to the server and vice-versa while reading and injecting data at will.



Packet Analysis of a Local Session Hijack

User
Server
SYN <Clt ISN 1200><WIN 512>
SYN <Svr ISN 1500><WIN 1024> /ACK 1201
ACK 1501
DATA=128 <Clt SEQ 1201>
ACK (Clt SEQ + DATA) 1329
DATA=91 <Clt SEQ 1329>
ACK (Clt SEQ + DATA) 1420
DATA=20<SEQ 1420>
ACK 1440
DATA=50<SEQ 1440>
Attacker

Note: Before the user sends the next data packet, the attacker predicts the next sequence number and sends the data to the server; this leads to the establishment of the connection between the attacker and the server

Packet Analysis of a Local Session Hijack

Session hijacking involves high-level attack vectors, which affect many systems. TCP is used for transmitting data by many systems that establish LAN or Internet connections. For establishing a connection between two systems and for the successful transmission of data, the two systems should perform a three-way handshake. Session hijacking involves the exploitation of this three-way handshake method to take control over the session.

To conduct a session hijacking attack, the attacker performs three activities:

- Tracking of a session
- Desynchronization of the session
- Injection of commands during the session

By sniffing network traffic, an attacker can monitor or track a session. The next step in session hijacking is to desynchronize the session. It is easy to accomplish this attack if the attacker knows the next sequence number (NSN) used by the client. A session can be hijacked by using that sequence number before the client uses it. There are two possibilities to determine sequence numbers: one is to sniff the traffic, find an ACK packet, and then determine the NSN based on the ACK packet. The other is to transmit data with guessed sequence numbers, which is not a reliable method. If the attacker can access the network and sniff the TCP session, they can easily determine the sequence number. This type of session hijacking is called "local session hijacking."

The below figure shows the packet analysis of a local session hijack.

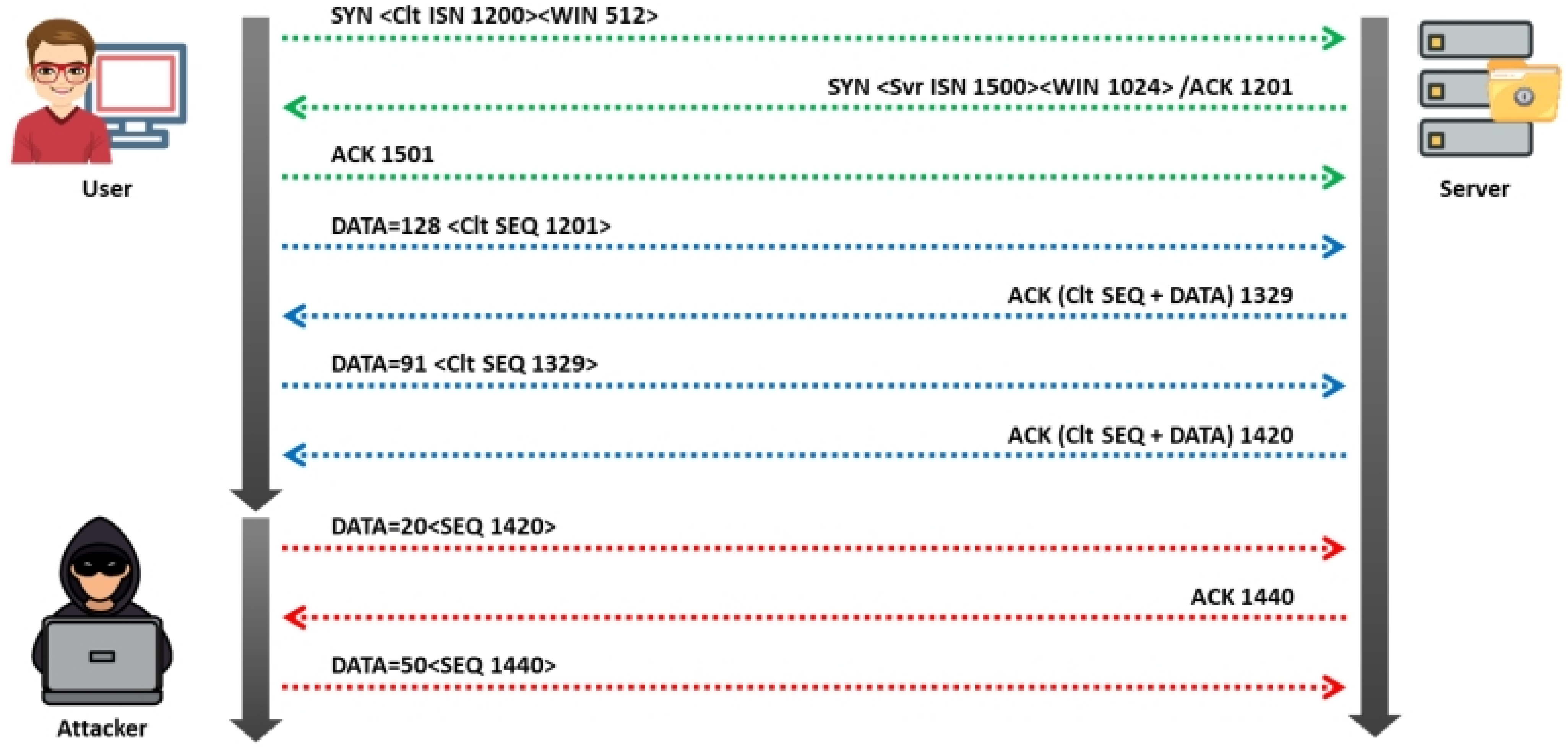
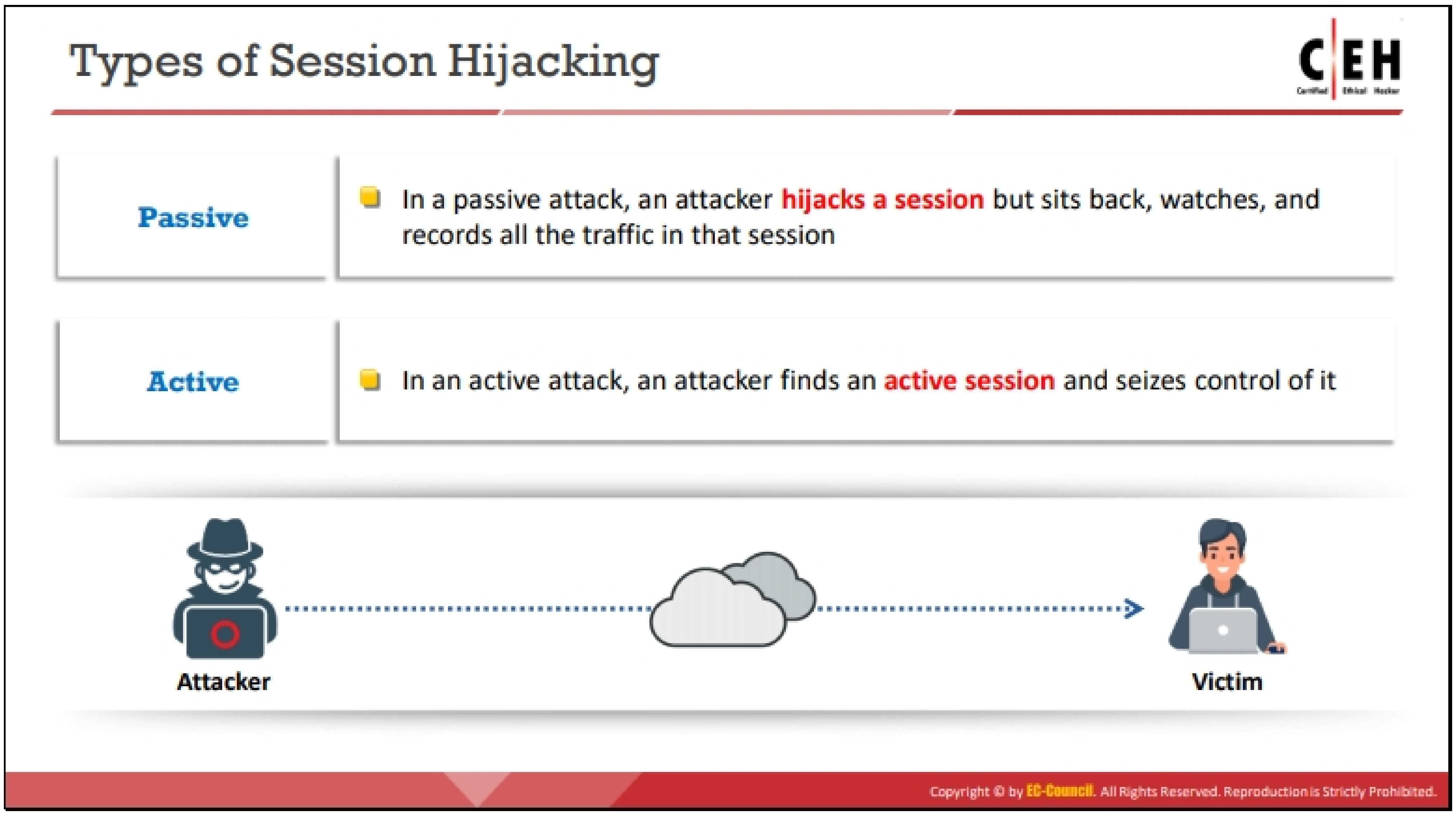

User
Server
Attacker
SYN <Clt ISN 1200><WIN 512>
SYN <Svr ISN 1500><WIN 1024> /ACK 1201
ACK 1501
DATA=128 <Clt SEQ 1201>
ACK (Clt SEQ + DATA) 1329
DATA=91 <Clt SEQ 1329>
ACK (Clt SEQ + DATA) 1420
DATA=20<SEQ 1420>
ACK 1440
DATA=50<SEQ 1440>

Figure 11.3: Packet analysis of a local session hijack

According to above figure, the next expected sequence number is 1420. If the attacker transmits that packet sequence number before the user does, they can desynchronize the connection between the user and server.

If the attacker sent the data with the expected sequence number before the user could, the server would be synchronized with the attacker. This leads to the establishment of a connection between the attacker and server. Then, the server would drop the data sent by the user with the correct sequence number, believing it to be a resent packet. The user is unaware of the attacker's action and may resend the data packet because the user does not receive an ACK for their TCP packet. However, the server would drop all the packets resent by the user. Thus, the local session hijacking attack is successfully completed.



Types of Session Hijacking

Passive	In a passive attack, an attacker **hijacks a session** but sits back, watches, and records all the traffic in that session
Active	In an active attack, an attacker finds an **active session** and seizes control of it

Attacker
Victim

Types of Session Hijacking

Session hijacking can be either active or passive, depending on the degree of involvement of the attacker. The essential difference between an active and passive hijack is that while an active hijack takes over an existing session, a passive hijack monitors an ongoing session.

- **Passive Session Hijacking**

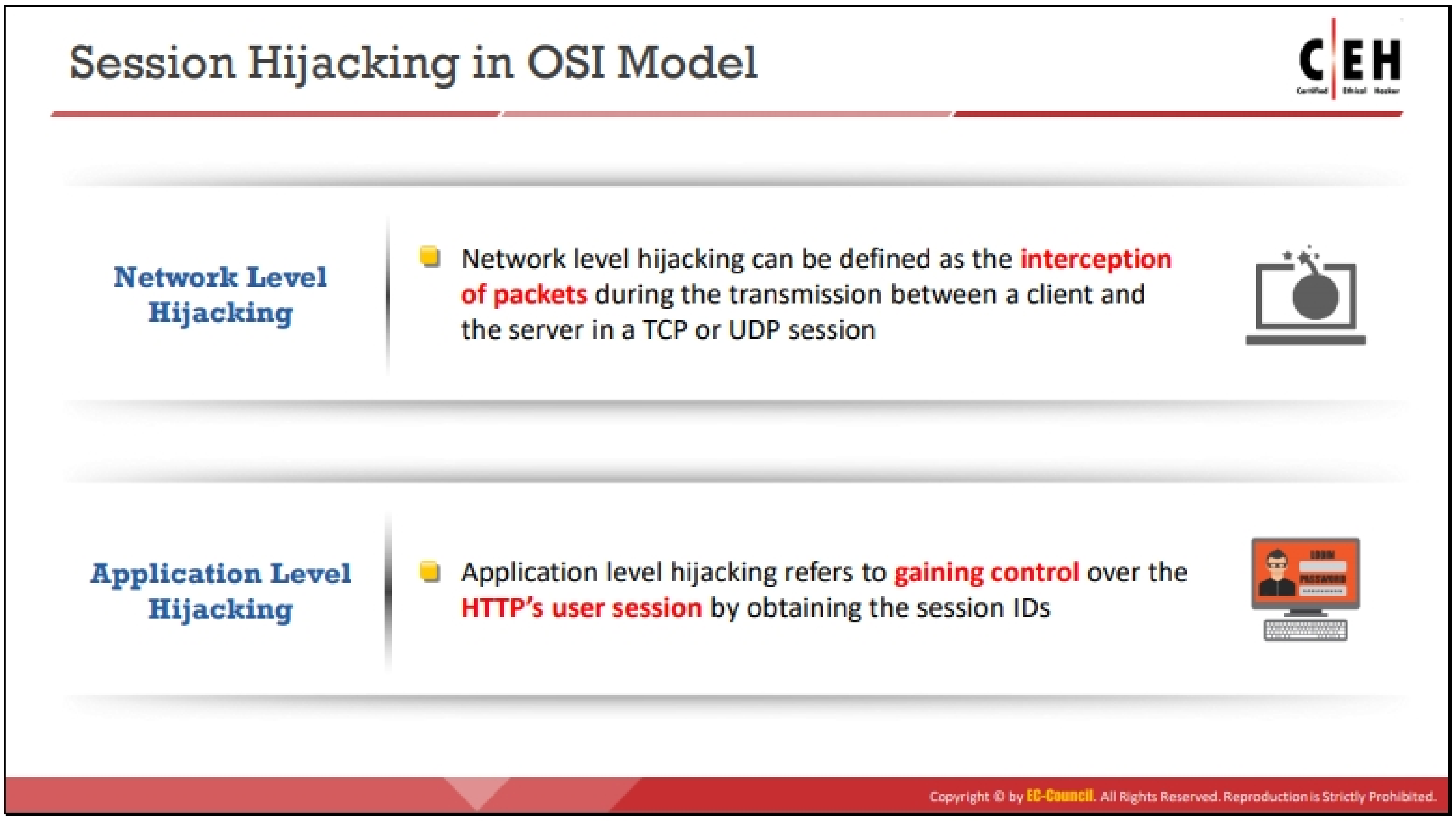
  In a passive attack, after hijacking a session, an attacker only observes and records all the traffic during the session. A passive attack uses sniffers on the network, allowing attackers to obtain information such as user IDs and passwords. The attacker can later use this information to log in as a valid user and enjoy the user's privileges. Password sniffing is the simplest attack to obtain raw access to a network. Countering this attack involves methods that range from identification schemes (for example, one-time password systems such as S/KEY) to ticketing identification (for example, Kerberos). These techniques help in protecting data from sniffing attacks, but they cannot protect against active attacks if the data are unencrypted or do not carry a digital signature.

- **Active Session Hijacking**

  In an active attack, an attacker takes over an existing session either by breaking the connection on one side of the conversation or by actively participating. An example of an active attack is a man-in-the-middle (MITM) attack. To perform a successful MITM attack, the attacker must guess the sequence number before the target responds to the server. On most current networks, sequence-number prediction does not work, because operating-system (OS) vendors use random values for the initial sequence number, which makes it difficult to predict sequence numbers.

Attacker
Victim

Figure 11.4: Attacker sniffing a victim's traffic



Session Hijacking in OSI Model

Network Level Hijacking	Network level hijacking can be defined as the **interception of packets** during the transmission between a client and the server in a TCP or UDP session
Application Level Hijacking	Application level hijacking refers to **gaining control** over the **HTTP's user session** by obtaining the session IDs

Session Hijacking in OSI Model

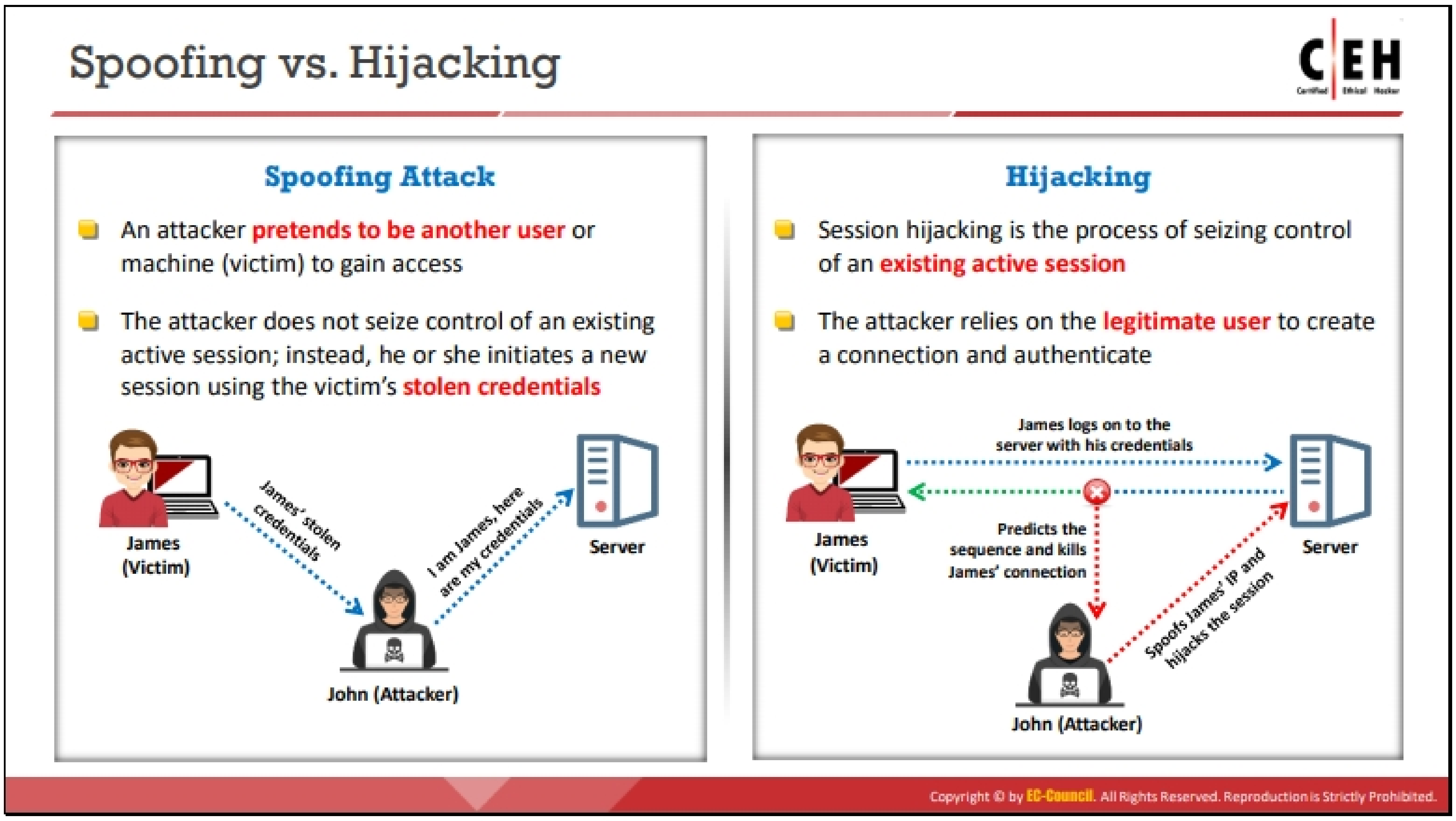
There are two levels of session hijacking in the OSI model: the network level and application level.

- **Network Level Hijacking**

  Network level hijacking is the interception of packets during the transmission between a client and server in a TCP/User Datagram Protocol (UDP) session. A successful attack provides the attacker with crucial information, which can be further used to attack application level sessions. Attackers most likely perform network level hijacking because they do not need to modify the attack on a per-web-application basis. This attack focuses on the data flow of the protocol shared across all web applications.

- **Application Level Hijacking**

  Application level hijacking involves gaining control over the Hypertext Transfer Protocol (HTTP) user session by obtaining the session IDs. At the application level, the attacker gains control of an existing session and can create new unauthorized sessions by using stolen data. In general, both occur together, depending on the system being attacked.



Spoofing vs. Hijacking

Spoofing Attack

- An attacker **pretends to be another user** or machine (victim) to gain access
- The attacker does not seize control of an existing active session; instead, he or she initiates a new session using the victim's **stolen credentials**

James (Victim)
James' stolen credentials
I am James, here are my credentials
Server
John (Attacker)

Hijacking

- Session hijacking is the process of seizing control of an **existing active session**
- The attacker relies on the **legitimate user** to create a connection and authenticate

James logs on to the server with his credentials
James (Victim)
Predicts the sequence and kills James' connection
Server
Spoofs James' IP and hijacks the session
John (Attacker)

Spoofing vs. Hijacking

In blind hijacking, an attacker predicts the sequence numbers that a victim host sends to create a connection that appears to originate from the host or a blind spoof. To understand blind hijacking, it is important to understand sequence-number prediction. TCP sequence numbers, which are unique per byte in a TCP session, provide flow control and data integrity. TCP segments provide the initial sequence number (ISN) as a part of each segment header. ISNs do not start at zero for each session. As part of the handshake process, each participant needs to state the ISN, and bytes are numbered sequentially from that point.

Blind session hijacking relies on the attacker's ability to predict or guess sequence numbers. An attacker is unable spoof a trusted host on a different network and observe the reply packets because no route exists for the packets to return to the attacker's IP address. Moreover, the attacker is unable resort to Address Resolution Protocol (ARP) cache poisoning because routers do not broadcast ARP across the Internet. Because the attacker is unable to observe the replies, they must anticipate the responses from the victim and prevent the host from sending a TCP/RST packet to the victim. The attacker predicts sequence numbers that the remote host expects from the victim and then hijacks the communication. This method is useful to exploit the trust relationships between users and remote machines.

In a spoofing attack, an attacker pretends to be another user or machine (victim) to gain access. Instead of taking over an existing active session, the attacker initiates a new session using the victim's stolen credentials. Simple IP spoofing is easy to perform and is useful in various attack methods. To create new raw packets, the attacker must have root access on the machine. However, to establish a spoofed connection using this session hijacking technique, an attacker must know the sequence numbers used by a target machine. IP spoofing forces the attacker to forecast the NSN. When an attacker uses blind hijacking to send a command, they cannot view the response.

In the case of IP spoofing without a session hijack, guessing the sequence number is unnecessary because no currently open session exists with that IP address. In a session hijack, the traffic returns to the attacker only if source routing is used. Source routing is a process that allows the sender to specify the route to be taken by an IP packet to the destination. The attacker performs source routing and then sniffs the traffic as it passes by the attacker. In session spoofing, captured authentication credentials are used to establish a session. In contrast, active hijacking eclipses a pre-existing session. As a result of this attack, a legitimate user may lose access or the normal functionality of their established Telnet session because an attacker hijacks the session and acts with the user's privileges. Because most authentication mechanisms are enforced only at the initiation of a session, the attacker can gain access to a target machine without authentication while a session is in progress.

Another method is to use source routed IP packets. This type of MITM attack allows an attacker to become a part of the target–host conversation by deceptively guiding IP packets to pass through their system.

Session hijacking is the process of taking over an existing active session. An attacker relies on a legitimate user to make a connection and authenticate. Session hijacking is more difficult than IP address spoofing. In session hijacking, John (an attacker) would seek to insert himself into a session that James (a legitimate user) already had set up with \\Mail. John would wait until James establishes a session, displace James from the established session by some means, such as a DoS attack, and then pick up the session as though he were James. Subsequently, John would send a scripted set of packets to \\Mail and observe the responses. For this purpose, John needs to know the sequence number in use when he hijacked the session. To calculate the sequence number, he must know the ISN and the number of packets involved in the exchange process.

Successful session hijacking is difficult without the use of known tools and is only possible when several factors are under the attacker's control. Knowledge of the ISN is the least of John's challenges. For instance, John needs a method to displace James from the active session as well as a method to know the exact status of James's session at the moment that James is displaced. Both these tasks require John to have far more knowledge and control over the session than would normally be possible.

However, IP address spoofing attacks can only be successful if an attacker uses IP addresses for authentication. They cannot perform IP address spoofing or session hijacking if per-packet integrity checking is executed. In the same manner, IP address spoofing or session hijacking is not possible if the session uses encryption methods such as Secure Sockets Layer (SSL) or Point-to-Point Tunneling Protocol (PPTP). Consequently, the attacker cannot participate in the key exchange.

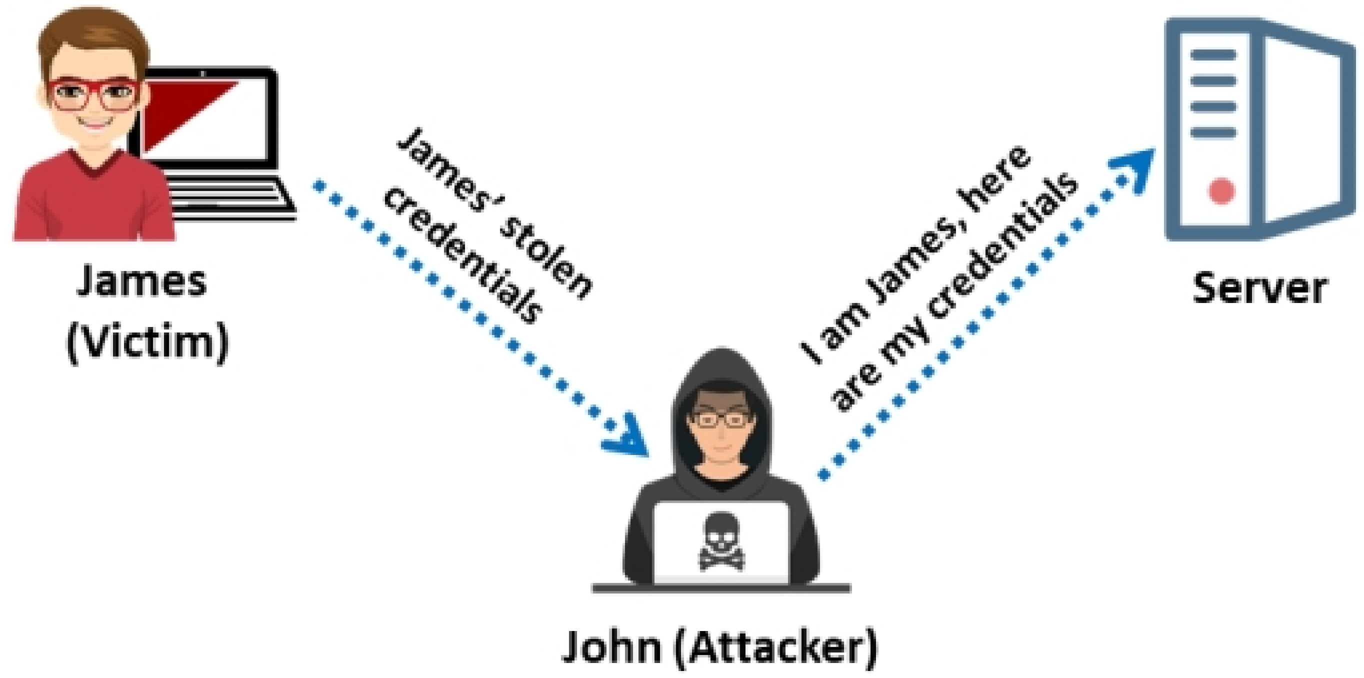

James
(Victim)
James' stolen credentials
I am James, here are my credentials
Server
John (Attacker)

Figure 11.5: Spoofing attack

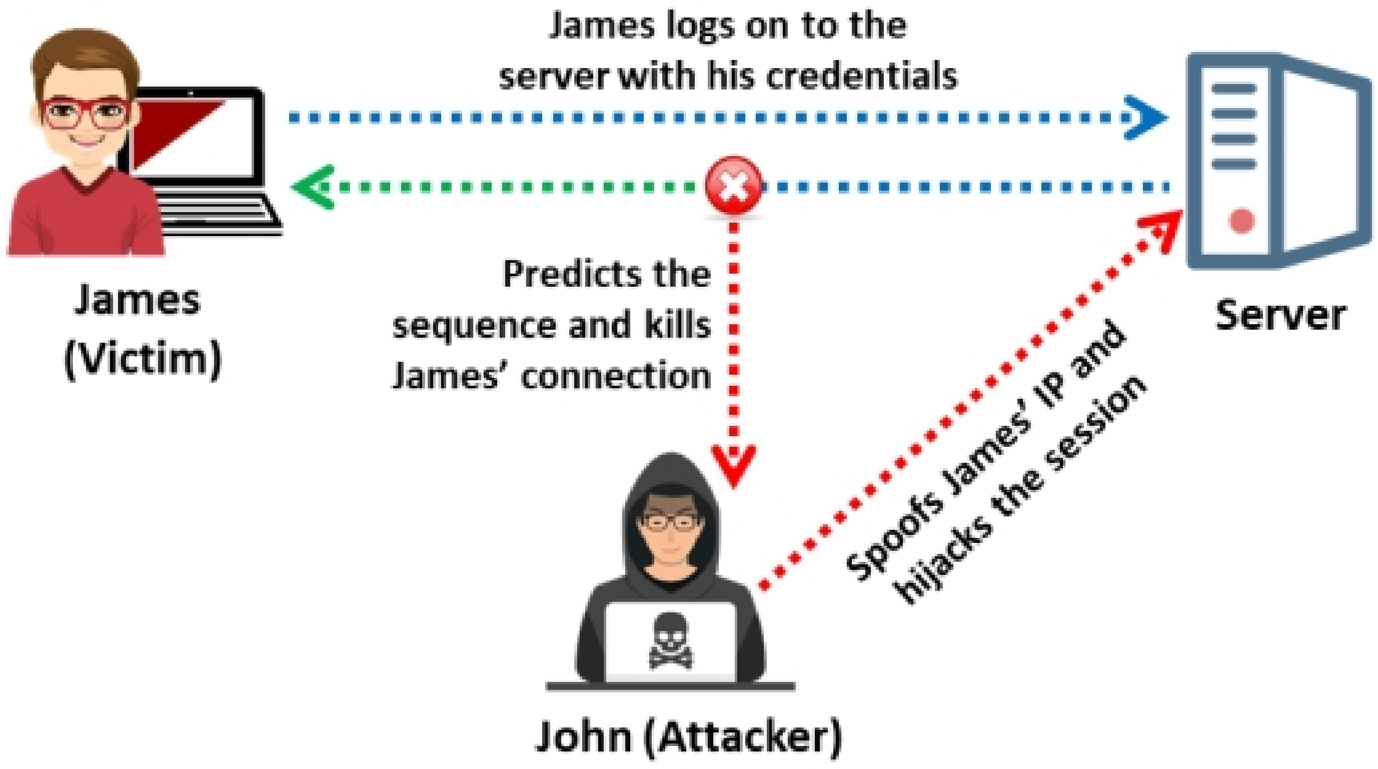
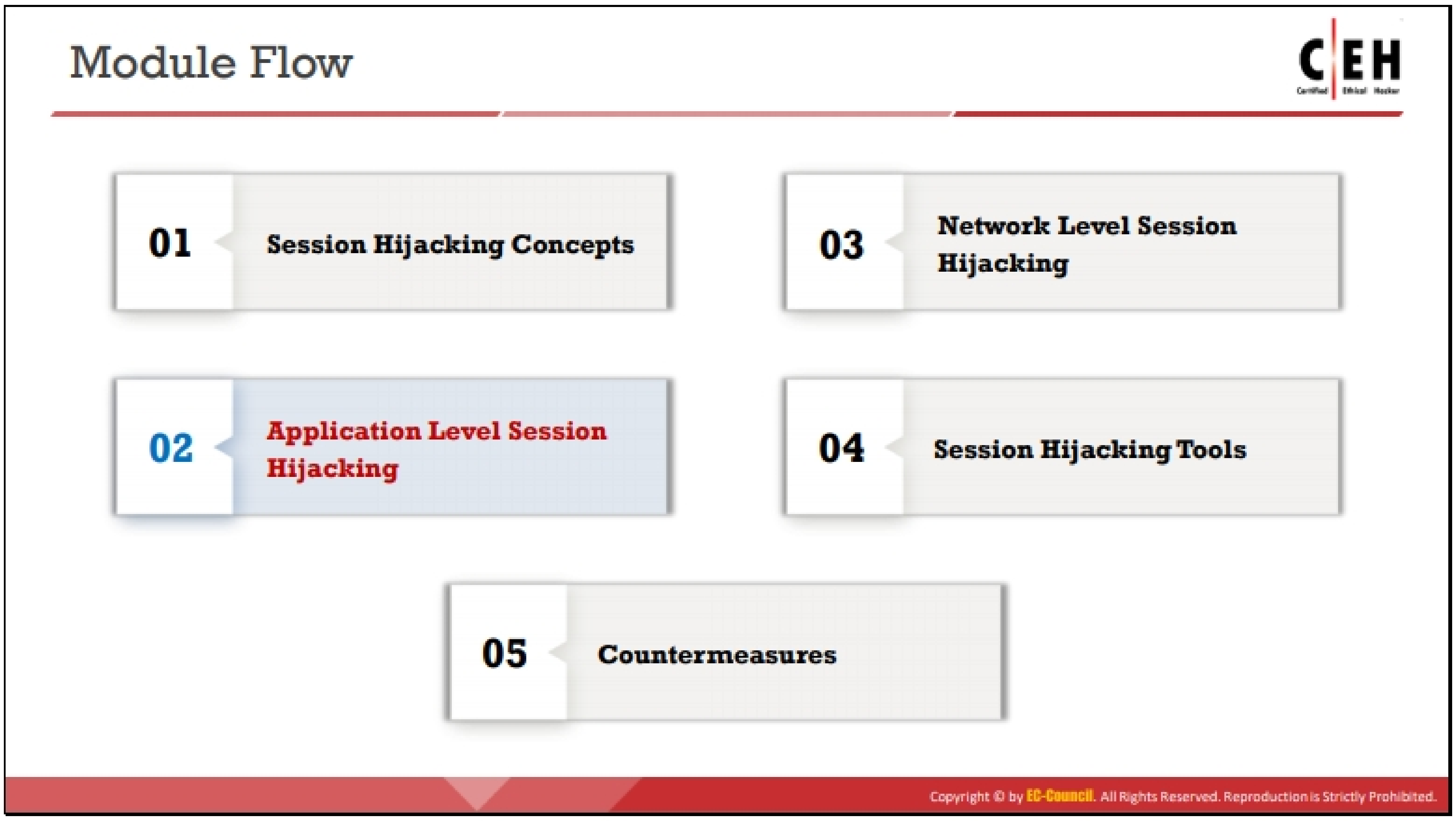

James logs on to the server with his credentials
James
(Victim)
Predicts the sequence and kills James' connection
Server
Spoofs James' IP and hijacks the session
John (Attacker)

Figure 11.6: Session hijacking

In summary, the hijacking of non-encrypted TCP communications requires the presence of non-encrypted session-oriented traffic, the ability to recognize TCP sequence numbers from which the next sequence number (NSN) can be predicted, and the ability to spoof a host's media access control (MAC) or IP address to receive communications that are not destined for the attacker's host. If the attacker is on the local segment, they can sniff and predict the ISN + 1 number and route the traffic back to them by poisoning the ARP caches on the two legitimate hosts participating in the session.



Module Flow

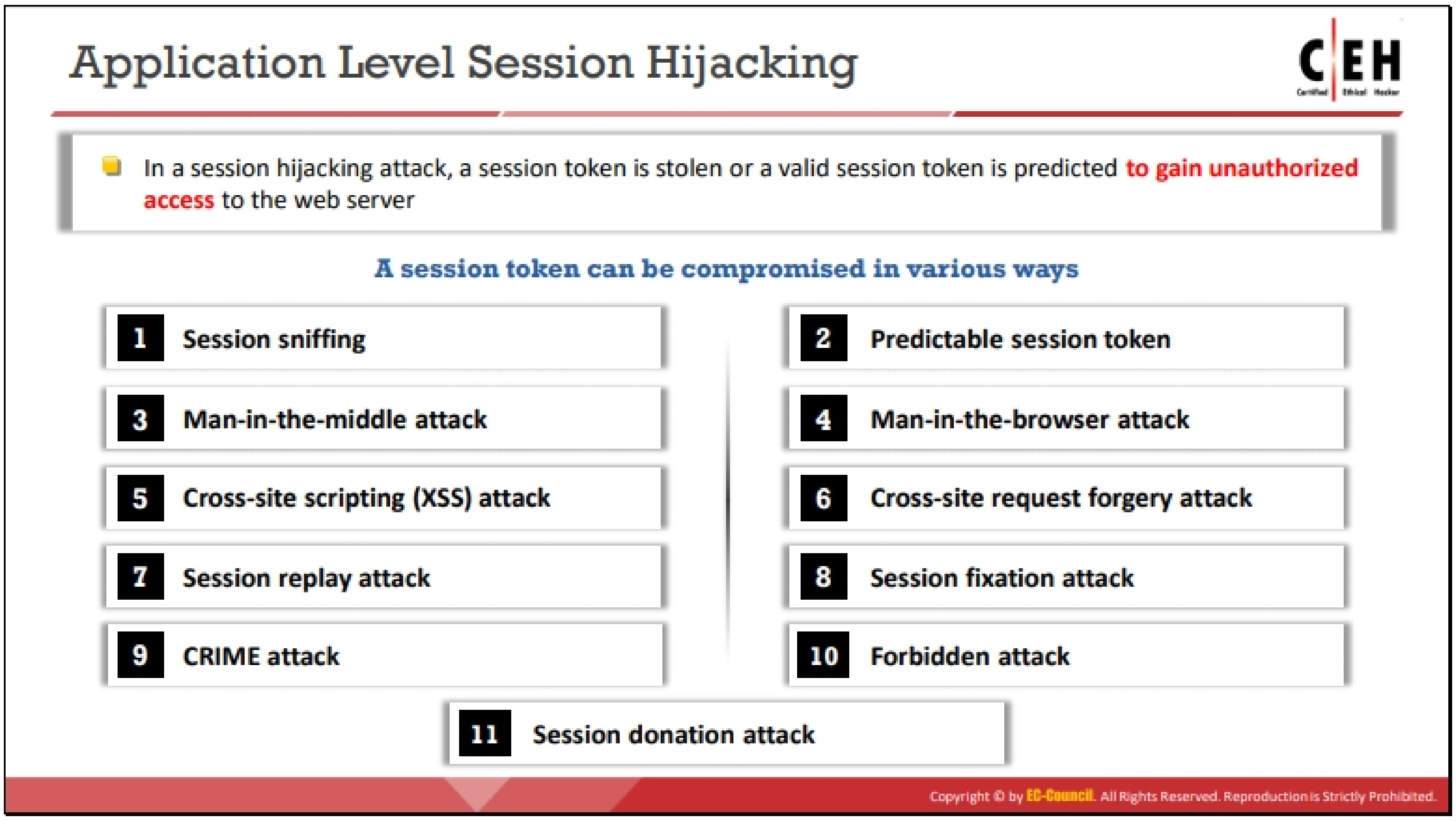
01 Session Hijacking Concepts

02 Application Level Session Hijacking

03 Network Level Session Hijacking

04 Session Hijacking Tools

05 Countermeasures



Application Level Session Hijacking

- In a session hijacking attack, a session token is stolen or a valid session token is predicted to gain unauthorized access to the web server

A session token can be compromised in various ways

1. Session sniffing
2. Predictable session token
3. Man-in-the-middle attack
4. Man-in-the-browser attack
5. Cross-site scripting (XSS) attack
6. Cross-site request forgery attack
7. Session replay attack
8. Session fixation attack
9. CRIME attack
10. Forbidden attack
11. Session donation attack

Application Level Session Hijacking

In application level hijacking, the attacker obtains the session IDs to gain control over an existing session or to create a new unauthorized session. This section discusses application level session hijacking and various methods to compromise the session token, such as session sniffing and the use of predictable session tokens.

In application level session hijacking, an attacker steals or predicts a valid session to gain unauthorized access to a web server. Usually, network level and application level session hijacking occur together because a successful network level session hijack provides an attacker with ample information to perform application level session hijacking. Application level session hijacking relies on HTTP sessions.

An attacker implements various techniques such as stealing, guessing, and brute forcing to obtain a valid session ID, which helps in acquiring control over a valid user's session while it is in progress.

- **Stealing**: Attackers use different techniques to steal session IDs. An attacker can steal the session key through physical access by, for example, acquiring the files containing session IDs or memory contents of either the user's system or the server. The attacker can also use sniffing tools such as Wireshark or SteelCentral Packet Analyzer to sniff the traffic between the client and server to extract the session IDs from the packets.
- **Guessing**: An attacker attempts to guess the session IDs by observing session variables. In the case of session hijacking, the range of session ID values that can be guessed is limited. Thus, guessing techniques are effective only when servers use weak or flawed session-ID generation mechanisms.
- **Brute forcing**: In the brute-force technique, an attacker obtains session IDs by attempting all possible permutations of session ID values until finding one that works. An attacker using a digital subscriber line (DSL) can generate up to 1,000 session IDs per second. This technique is most useful when the algorithm that produces session IDs is non-random.

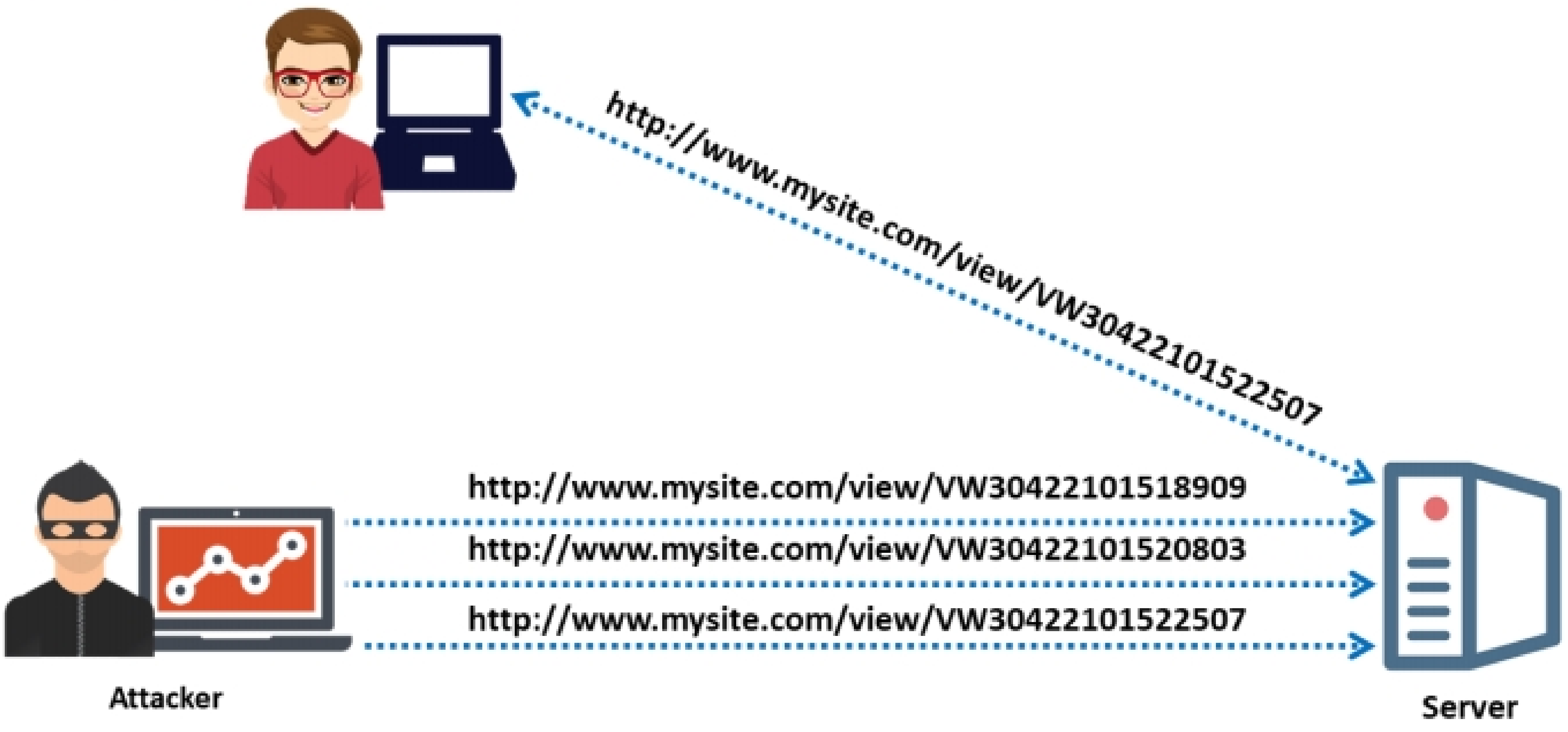

http://www.mysite.com/view/VW30422101522507
http://www.mysite.com/view/VW30422101518909
http://www.mysite.com/view/VW30422101520803
http://www.mysite.com/view/VW30422101522507
Attacker
Server

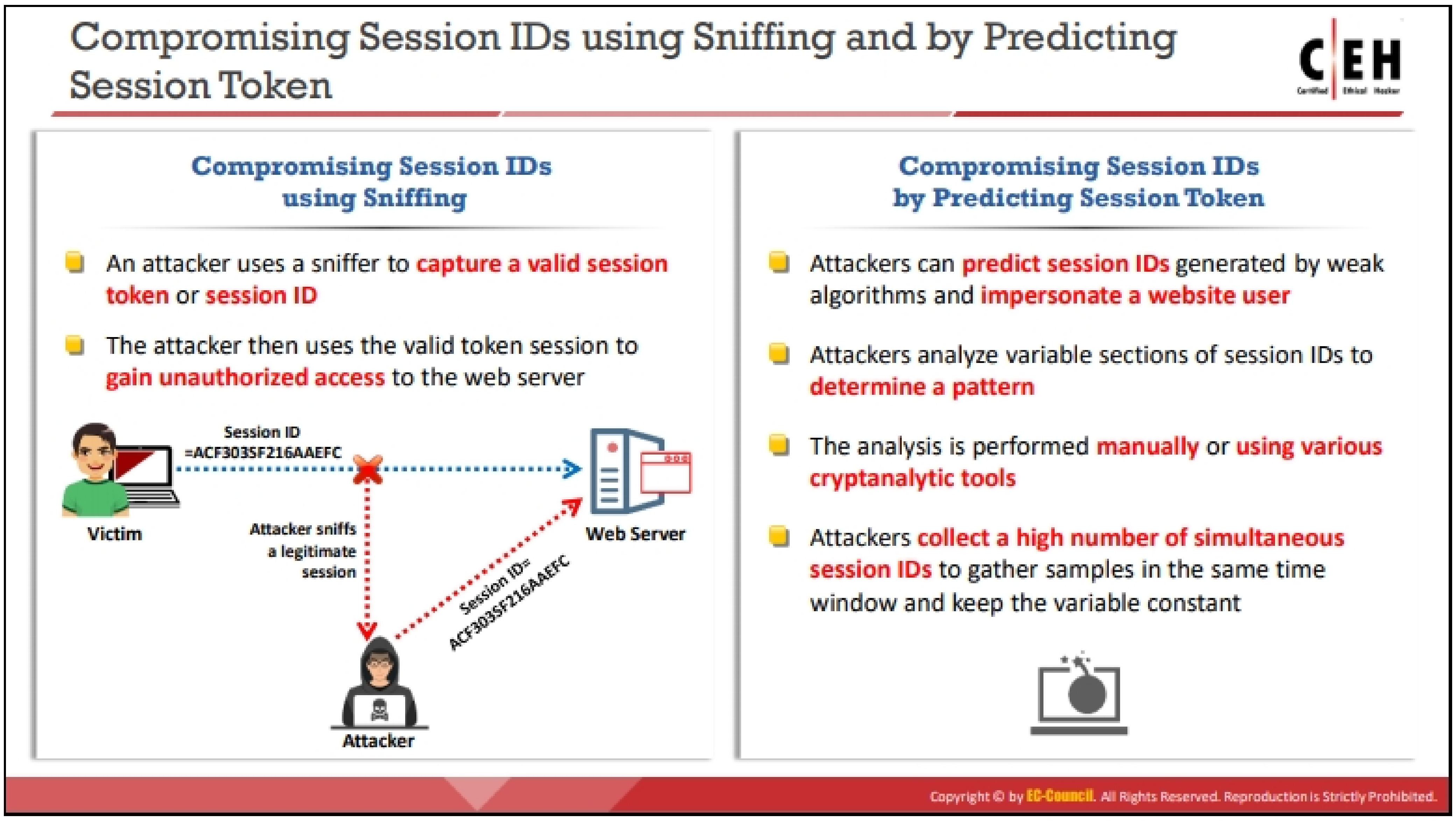
Figure 11.7: Brute-forcing attack on the session ID of a user

As shown in the above figure, a legitimate user connects to a server with session ID VW30422101522507. Employing various combinations such as VW30422101518909 and VW30422101520803, an attacker attempts to brute force the session ID in the hope of eventually arriving at the correct session ID. Once the attacker obtains the correct session ID, they gain complete access to the user's data and can perform operations on behalf of the legitimate user.

Note: A session ID brute-forcing attack is known as a session prediction attack if the predicted range of values for a session ID is very small.

A session token can be compromised in various ways:

- Session sniffing
- Predictable session token
- Man-in-the-middle (MITM) attack
- Man-in-the-browser attack
- Cross-site scripting (XSS) attack
- Cross-site request forgery attack
- Session replay attack
- Session fixation attack
- CRIME attack
- Forbidden attack
- Session donation attack



Compromising Session IDs using Sniffing and by Predicting Session Token

Compromising Session IDs using Sniffing

- An attacker uses a sniffer to **capture a valid session token** or **session ID**
- The attacker then uses the valid token session to **gain unauthorized access** to the web server

Session ID
=ACF303SF216AAEFC
Victim
Attacker sniffs a legitimate session
Web Server
Session ID=
ACF303SF216AAEFC
Attacker

Compromising Session IDs by Predicting Session Token

- Attackers can **predict session IDs** generated by weak algorithms and **impersonate a website user**
- Attackers analyze variable sections of session IDs to **determine a pattern**
- The analysis is performed **manually** or **using various cryptanalytic tools**
- Attackers **collect a high number of simultaneous session IDs** to gather samples in the same time window and keep the variable constant

Compromising Session IDs Using Sniffing

A web server identifies a user's connection through a unique session ID (also known as a session token). The web server sends a session token to the client browser after the successful authentication of client login. Usually, a session token comprises a string of variable width that is useful in various ways, such as in the header of an HTTP requisition (cookie), in a URL, or in the body of an HTTP requisition.

An attacker uses packet sniffing tools such as Wireshark and SteelCentral Packet Analyzer to intercept the HTTP traffic between a victim and web server. The attacker then analyzes the data in the captured packets to identify valuable information such as session IDs and passwords. Once the session ID is determined, the attacker masquerades as the victim and sends the session ID to the web server before the victim does. The attacker uses the valid token session to gain unauthorized access to the web server. In this manner, the attacker takes control over an existing legitimate session.

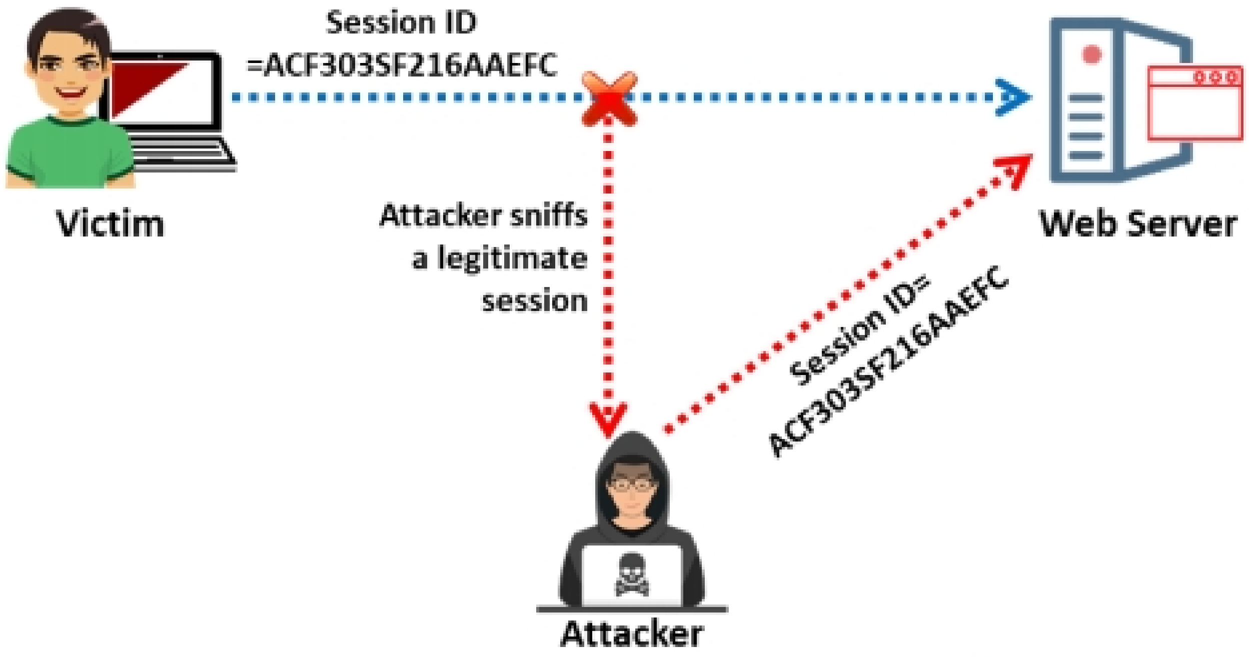

Session ID
=ACF303SF216AAEFC
Victim
Attacker sniffs
a legitimate
session
Web Server
Session ID=
ACF303SF216AAEFC
Attacker

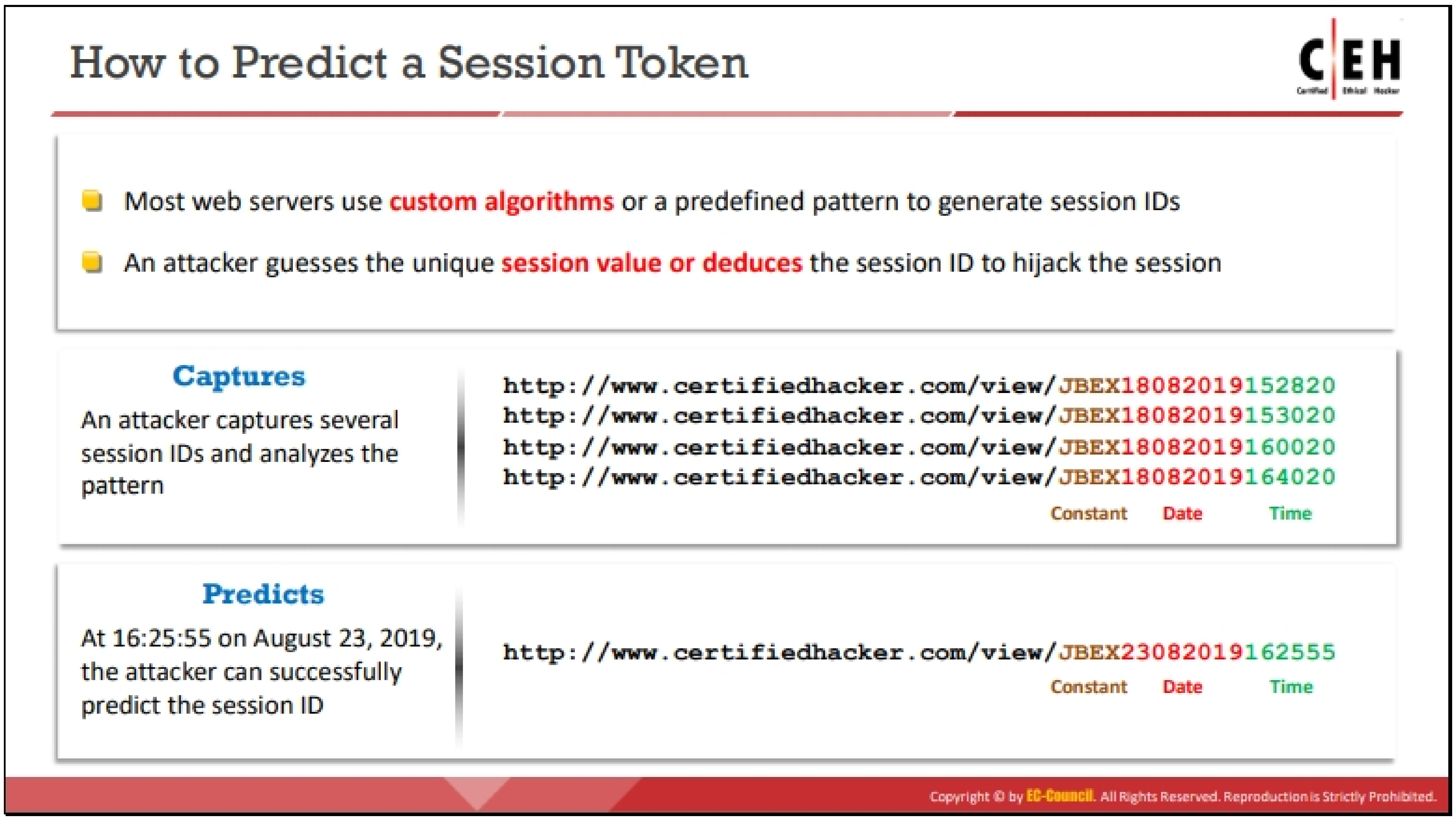
Figure 11.8: Prediction of session ID by sniffing

Compromising Session IDs by Predicting Session Token

A session ID is tagged as proof of an authenticated session established between a user and web server. Thus, if an attacker can guess or predict the session ID of the user, fraudulent activity is possible. Session prediction enables an attacker to bypass the authentication schema of an application. Usually, attackers can predict session IDs generated by weak algorithms and impersonate a website user. Attackers analyze a variable section of session IDs to determine the existence of a pattern. This analysis is performed either manually or by using various cryptanalytic tools.

An attacker collect a high number of simultaneous session IDs to gather samples in the same time window and keep the variable constant. First, the attacker collects some valid session IDs that are useful in identifying authenticated users. The attacker then studies the session ID structure, the information used to generate it, and the algorithm used by the web application to secure it. From these findings, the attacker can predict the session ID.

Attackers can also guess session IDs by using a brute-force technique, in which they generate and test different session ID values until they succeed in gaining access to the application.



How to Predict a Session Token

- Most web servers use **custom algorithms** or a predefined pattern to generate session IDs
- An attacker guesses the unique **session value or deduces** the session ID to hijack the session

Captures

An attacker captures several session IDs and analyzes the pattern

```
http://www.certifiedhacker.com/view/JBEX18082019152820
http://www.certifiedhacker.com/view/JBEX18082019153020
http://www.certifiedhacker.com/view/JBEX18082019160020
http://www.certifiedhacker.com/view/JBEX18082019164020
```

Constant Date Time

Predicts

At 16:25:55 on August 23, 2019, the attacker can successfully predict the session ID

```
http://www.certifiedhacker.com/view/JBEX23082019162555
```

Constant Date Time

How to Predict a Session Token

Most web servers generate session IDs using custom algorithms or a pre-defined pattern that might simply increase static numbers, whereas others use more complex procedures such as factoring in time and other computer-specific variables. Thus, attackers can identify session IDs generated in the following ways:

- Embedding in the URL, which is received by a GET request in the application when the links embedded within a page are clicked by clients
- Embedding in a form as a hidden field, which is submitted to the HTTP's POST command
- Embedding in cookies on the client's local machine

An attacker guesses the unique session value or deduces the session ID to hijack the session. As shown in the below figure, an attacker first captures several session IDs and analyzes the pattern.

```
http://www.certifiedhacker.com/view/JBEX18082019152820
http://www.certifiedhacker.com/view/JBEX18082019153020
http://www.certifiedhacker.com/view/JBEX18082019160020
http://www.certifiedhacker.com/view/JBEX18082019164020
```

Constant Date Time

Figure 11.9: Sample sessions captured by an attacker

On analyzing the pattern, at 16:25:55 on August 23, 2019, the attacker successfully predicts the session ID, as shown in the below figure.

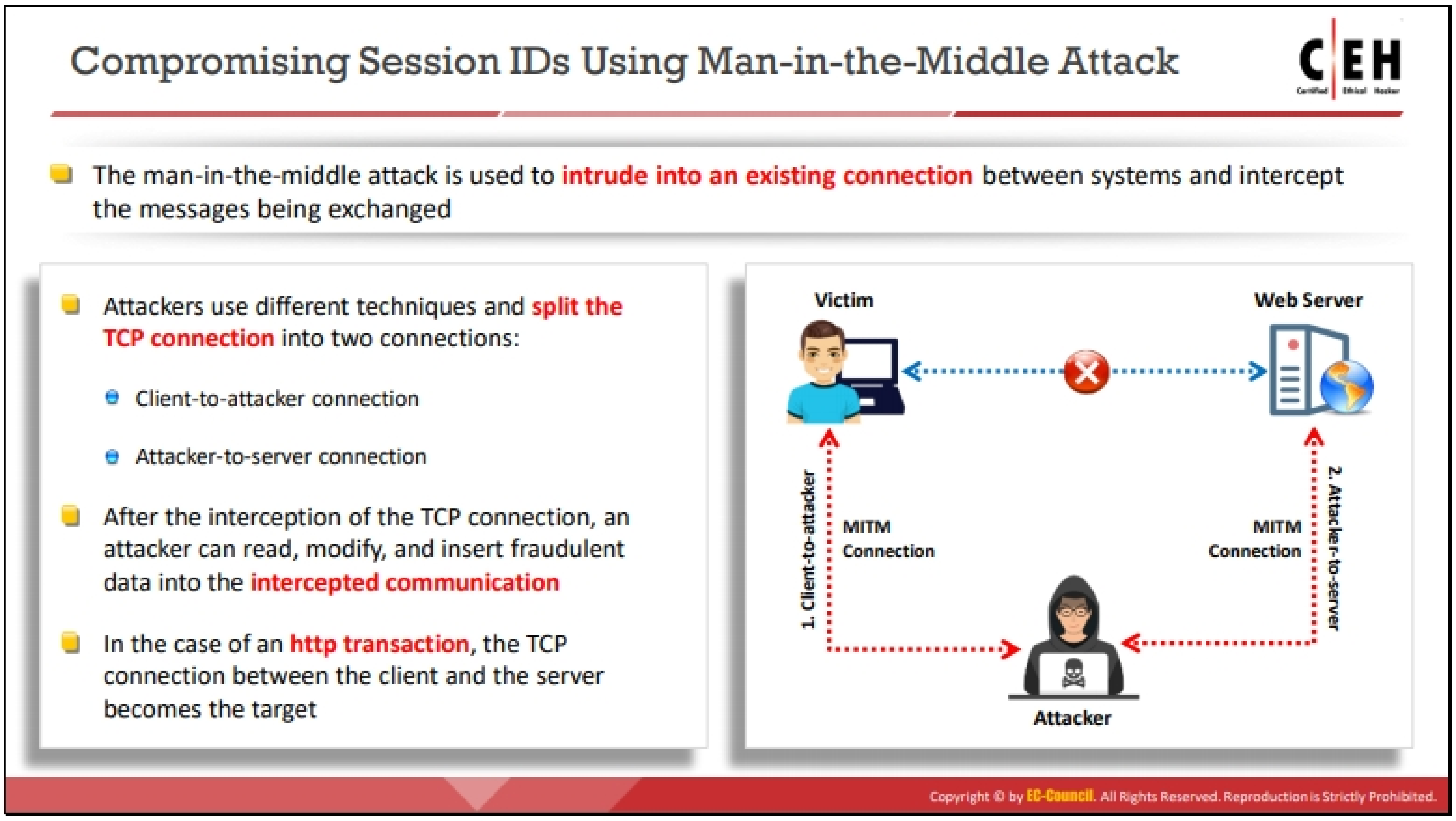
http://www.certifiedhacker.com/view/JBEX23082019162555

Constant **Date** **Time**

Figure 11.10: Session ID predicted by the attacker

Now, the attacker can mount an attack through the following steps.

- The attacker acquires the current session ID and connects to the web application.
- The attacker implements a brute-force technique or calculates the next session ID.
- The attacker modifies the current value in the cookie/URL/hidden form field and assumes the next user's identity.



Compromising Session IDs Using Man-in-the-Middle Attack

- The man-in-the-middle attack is used to **intrude into an existing connection** between systems and intercept the messages being exchanged

- Attackers use different techniques and **split the TCP connection** into two connections:
  - Client-to-attacker connection
  - Attacker-to-server connection
- After the interception of the TCP connection, an attacker can read, modify, and insert fraudulent data into the **intercepted communication**
- In the case of an **http transaction**, the TCP connection between the client and the server becomes the target

Victim
Web Server
1. Client-to-attacker
MITM Connection
MITM Connection
2. Attacker-to-server
Attacker

Compromising Session IDs Using Man-in-the-Middle Attack

A man-in-the-middle (MITM) attack is used to intrude into an existing connection between systems and to intercept messages being transmitted. In this attack, attackers use different techniques and split a TCP connection into two: a client-to-attacker connection and an attacker-to-server connection. After the successful interception of a TCP connection, an attacker can read, modify, and insert fraudulent data into the intercepted communication. In the case of an HTTP transaction, the TCP connection between the client and server is the target.

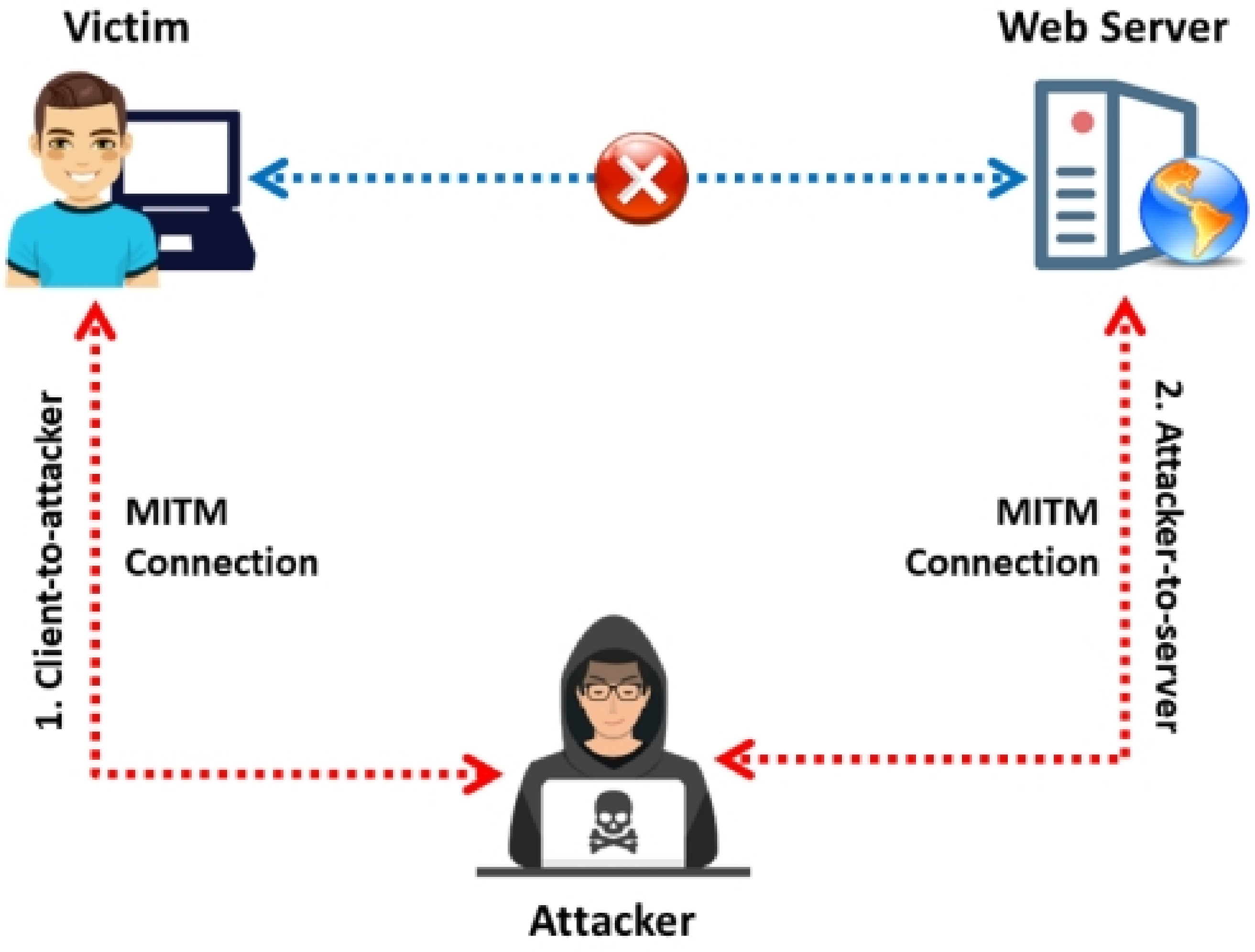
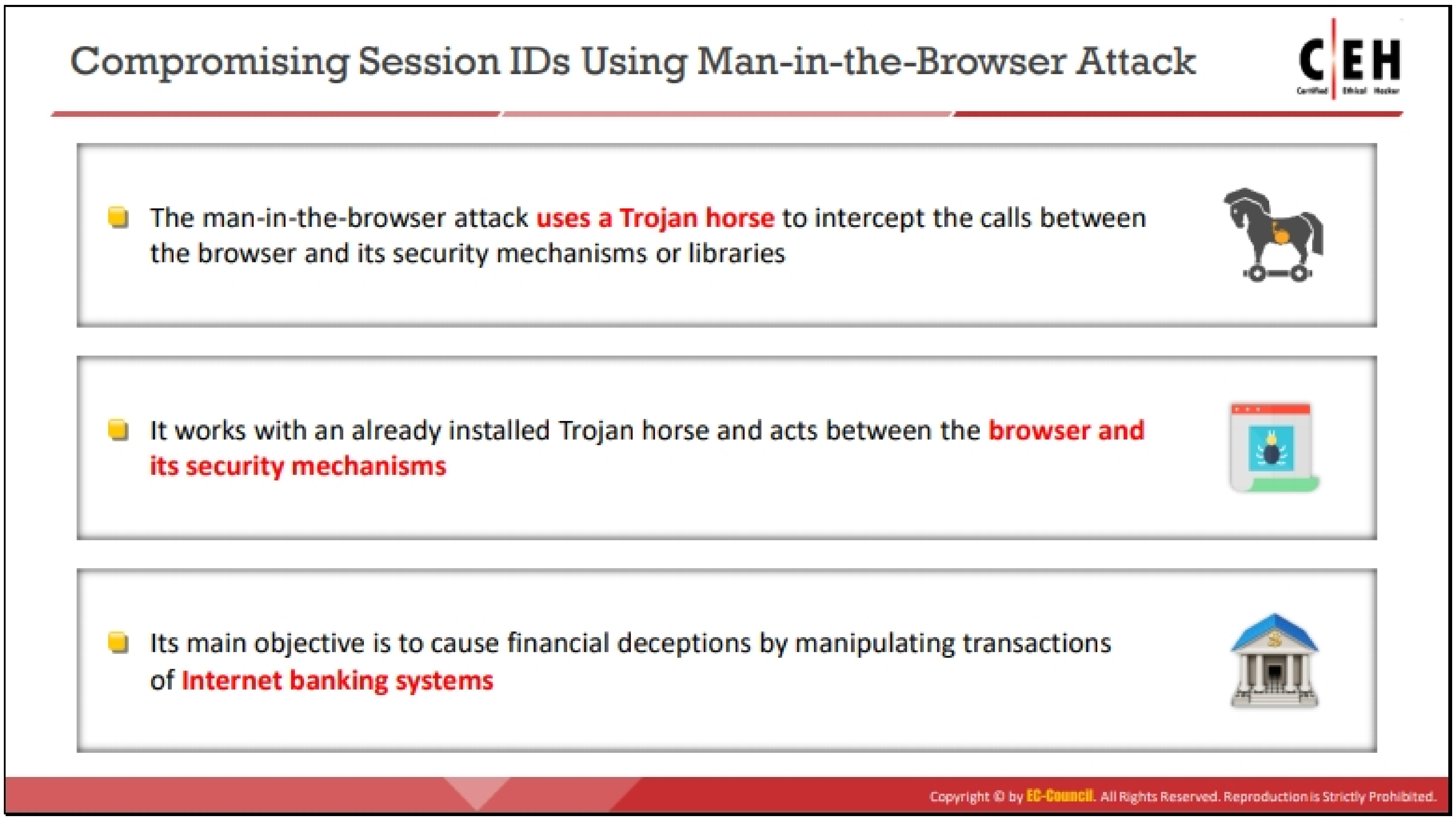
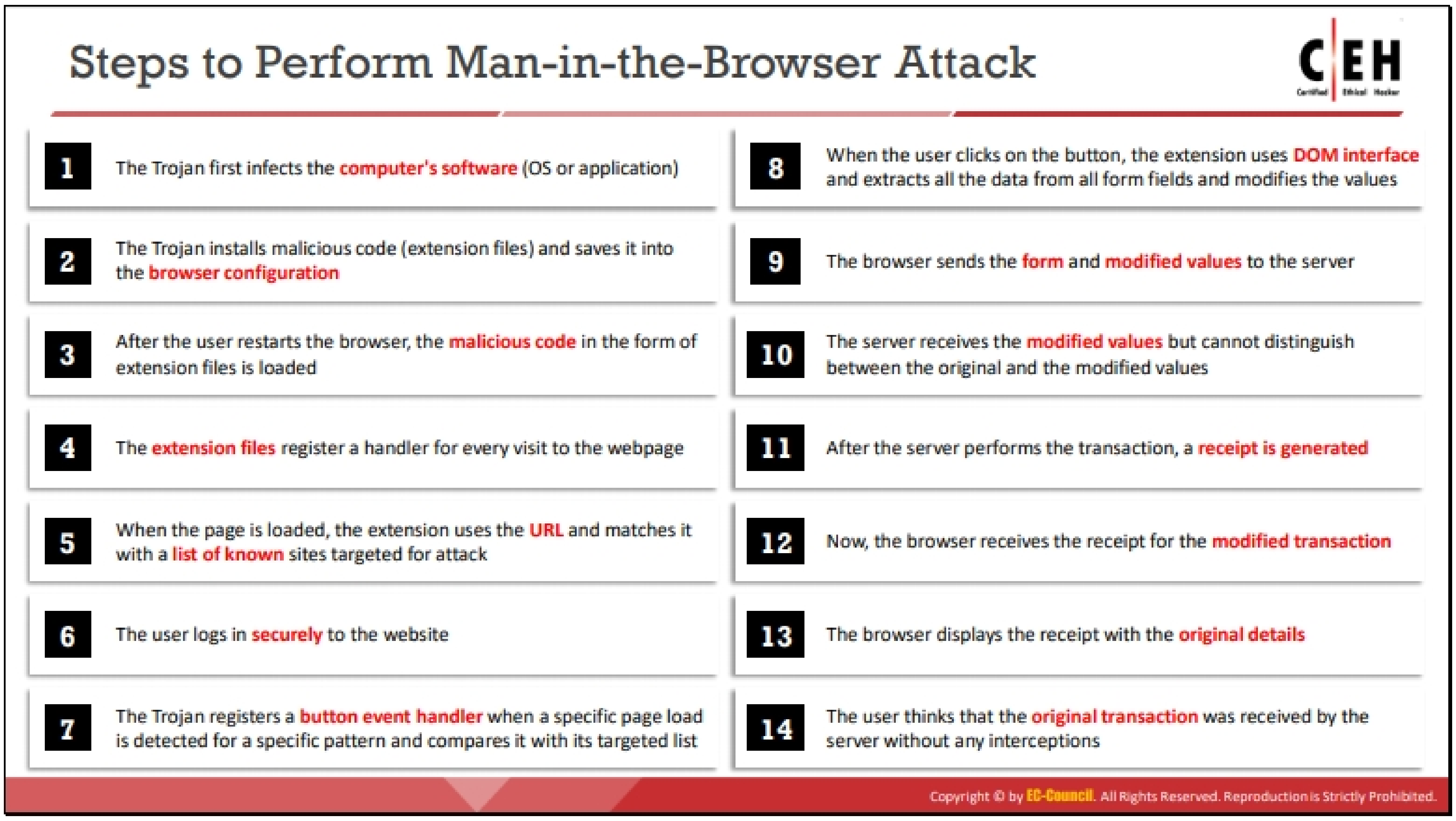

Victim
Web Server
1. Client-to-attacker
MITM Connection
MITM Connection
2. Attacker-to-server
Attacker

Figure 11.11: Prediction of session ID using a man-in-the-middle (MITM) attack



Compromising Session IDs Using Man-in-the-Browser Attack

- The man-in-the-browser attack **uses a Trojan horse** to intercept the calls between the browser and its security mechanisms or libraries
- It works with an already installed Trojan horse and acts between the **browser and its security mechanisms**
- Its main objective is to cause financial deceptions by manipulating transactions of **Internet banking systems**



Steps to Perform Man-in-the-Browser Attack

1. The Trojan first infects the **computer's software** (OS or application)
2. The Trojan installs malicious code (extension files) and saves it into the **browser configuration**
3. After the user restarts the browser, the **malicious code** in the form of extension files is loaded
4. The **extension files** register a handler for every visit to the webpage
5. When the page is loaded, the extension uses the **URL** and matches it with a **list of known** sites targeted for attack
6. The user logs in **securely** to the website
7. The Trojan registers a **button event handler** when a specific page load is detected for a specific pattern and compares it with its targeted list
8. When the user clicks on the button, the extension uses **DOM interface** and extracts all the data from all form fields and modifies the values
9. The browser sends the **form** and **modified values** to the server
10. The server receives the **modified values** but cannot distinguish between the original and the modified values
11. After the server performs the transaction, a **receipt is generated**
12. Now, the browser receives the receipt for the **modified transaction**
13. The browser displays the receipt with the **original details**
14. The user thinks that the **original transaction** was received by the server without any interceptions

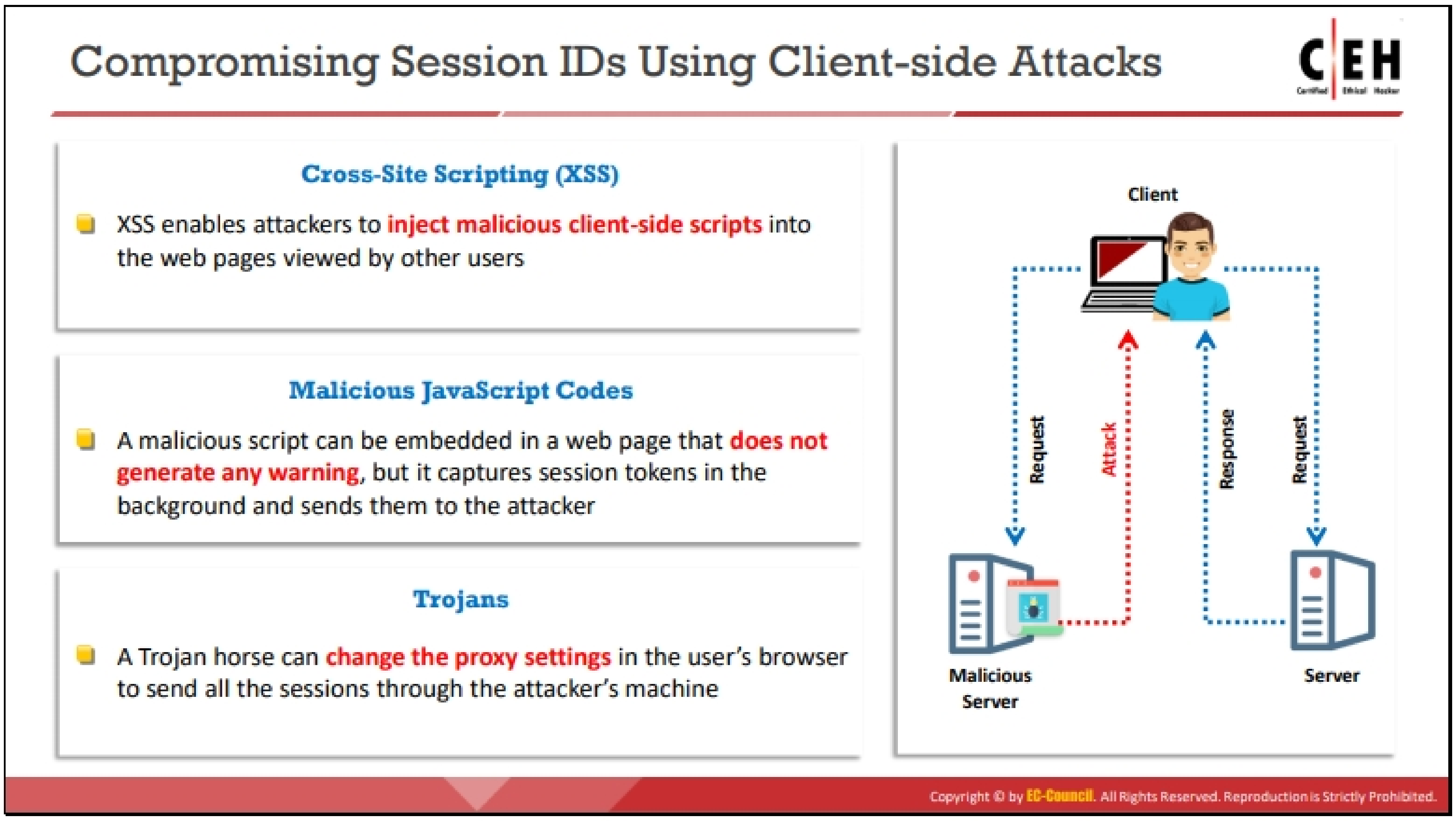
Compromising Session IDs Using Man-in-the-Browser Attack

A man-in-the-browser attack is similar to an MITM attack. The difference between the two is that a man-in-the-browser attack uses a Trojan horse to intercept and manipulate calls between a browser and its security mechanisms or libraries. An attacker positions a previously installed Trojan between the browser and its security mechanism, and the Trojan can modify web pages and transaction content or insert additional transactions. All of the Trojan's activities are invisible to both the user and web application.

The main objective of this attack is financial theft by manipulating transactions made using Internet banking systems. A man-in-the-browser attack can succeed even in the presence of security mechanisms such as SSL, public key infrastructure (PKI), and two-factor authentication because all the expected controls and security mechanisms would seem to function normally.

Steps to Perform Man-in-the-Browser Attack:

- The Trojan first infects the computer's software (OS or application).
- The Trojan installs malicious code (extension files) and saves it in the browser configuration.
- After the user restarts the browser, the malicious code in the form of extension files is loaded.
- The extension files register a handler for every visit to a webpage.
- When a page is loaded, the extension matches its URL with a list of known sites targeted for attack.
- The user logs in securely to the website.
- The extension registers a button event handler when a specific page load is detected with a specific pattern and compares it with its targeted list.
- When the user clicks on the button, the extension uses the Document Object Model (DOM) interface and extracts all the data from all form fields and modifies the values.
- The browser sends the form and modified values to the server.
- The server receives the modified values but cannot distinguish between the original and modified values.
- After the server performs the transaction, a receipt is generated.
- Now, the browser receives the receipt for the modified transaction.
- The browser displays the receipt with the original details.
- The user believes that the original transaction was received by the server without any interception.



Compromising Session IDs Using Client-side Attacks

Cross-Site Scripting (XSS)

- XSS enables attackers to **inject malicious client-side scripts** into the web pages viewed by other users

Malicious JavaScript Codes

- A malicious script can be embedded in a web page that **does not generate any warning**, but it captures session tokens in the background and sends them to the attacker

Trojans

- A Trojan horse can **change the proxy settings** in the user's browser to send all the sessions through the attacker's machine

Compromising Session IDs Using Client-side Attacks

Client-side attacks target vulnerabilities in client applications that interact with a malicious server or process malicious data. Depending on the nature of vulnerabilities, an attacker can exploit an application by sending an email with a malicious link or otherwise tricking a user into visiting a malicious website. Vulnerable client-side applications include unprotected websites, Java Runtime Environment, and browsers; of these, browsers are the major target. Client-side attacks occur when clients establish connections with malicious servers and process potentially harmful data from them. If no interaction occurs between the client and server, then there is no scope for a client-side attack. One such example is running a File Transfer Protocol (FTP) client without establishing a connection to an FTP server. In the case of instant messaging, the application is configured in such a way that it makes clients to log in to a remote server, making it susceptible to client-side attacks. The following client-side attacks can be used to compromise session IDs.

- **Cross-site scripting (XSS)**: XSS enables attackers to inject malicious client-side scripts into web pages viewed by other users.
- **Malicious JavaScript codes**: An attacker can embed in a web page a malicious script that does not generate any warning but captures session tokens in the background and sends them to the attacker.
- **Trojans**: A Trojan horse can change the proxy settings in the user's browser to send all sessions through an attacker's machine.

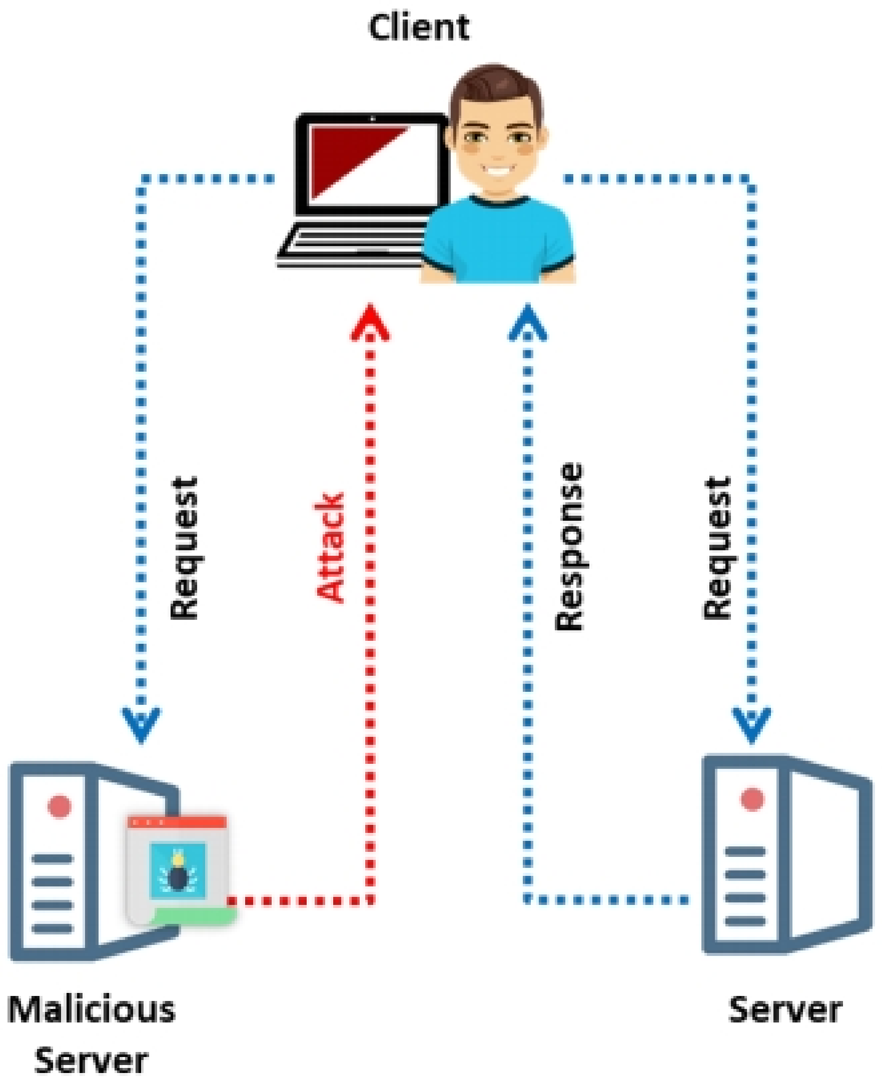
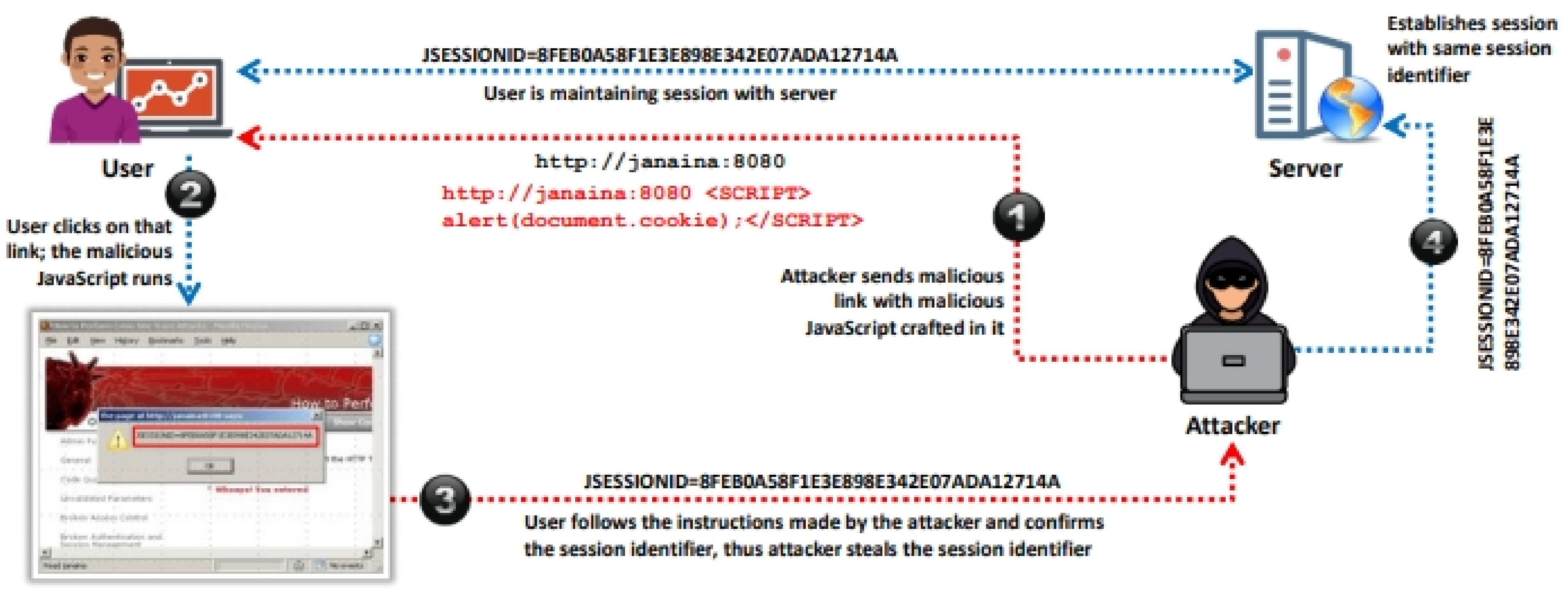

Client
Request
Attack
Response
Request
Malicious Server
Server

Figure 11.12: Prediction of session ID using a client-side attack

Compromising Session IDs Using Client-side Attacks: Cross-site Script Attack

- If an attacker sends a **crafted link** to the victim with **malicious JavaScript**, the JavaScript will run and complete the instructions made by the attacker when the victim clicks on the link


JSESSIONID=8FEB0A58F1E3E898E342E07ADA12714A
User is maintaining session with server
User
Server
Establishes session with same session identifier
http://janaina:8080
http://janaina:8080 <SCRIPT>
alert(document.cookie);</SCRIPT>
1
Attacker sends malicious link with malicious JavaScript crafted in it
2
User clicks on that link; the malicious JavaScript runs
Attacker
4
JSESSIONID=8FEB0A58F1E3E 898E342E07ADA12714A
3
JSESSIONID=8FEB0A58F1E3E898E342E07ADA12714A
User follows the instructions made by the attacker and confirms the session identifier, thus attacker steals the session identifier

Compromising Session IDs Using Client-side Attacks: Cross-site Script Attack

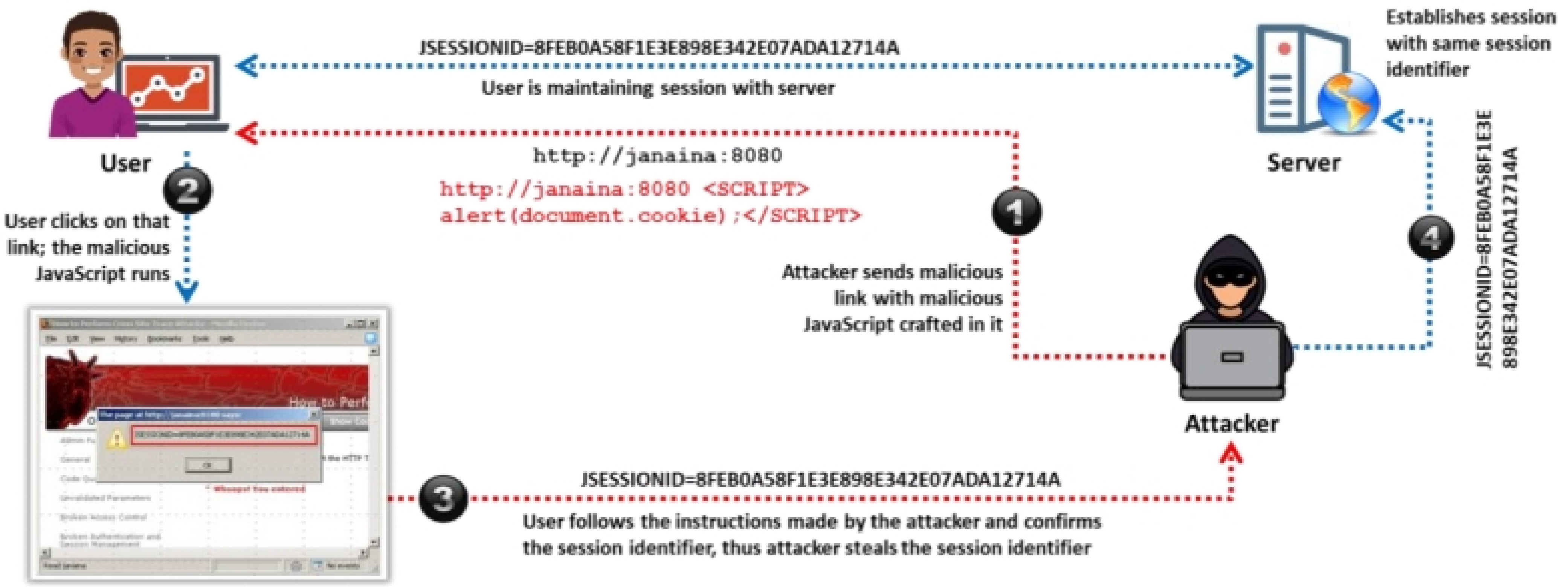
A cross-site script attack is a client-side attack in which the attacker compromises a session token by using malicious code or programs. This type of attack occurs when a dynamic web page receives malicious data from the attacker and executes it on the user's system.

Web sites that create dynamic pages do not have control over how the clients read their output. Thus, attackers can insert a malicious JavaScript, VBScript, ActiveX, Hypertext Markup Language (HTML), or Flash applet into a vulnerable dynamic page. That page then executes the script on the user's machine and collects personal information of the user, steals cookies, redirects users to unexpected web pages, or executes any malicious code on the user's system.

As shown in the below figure, a user first establishes a valid session with a server. An attacker sends a crafted link to the victim with malicious JavaScript. When the user clicks on the link, the JavaScript runs automatically and performs the instructions set by the attacker. The result displays the current session ID of the user. Using the same technique, the attacker can create specific JavaScript code that fetches the user's session ID:

```
<SCRIPT>alert(document.cookie);</SCRIPT>
```

Thereafter, the attacker uses the stolen session ID to establish a valid session with the server.


JSESSIONID=8FEB0A58F1E3E898E342E07ADA12714A
User
Server
Attacker
1
2
3
4
JSESSIONID=8FEB0A58F1E3E898E342E07ADA12714A
JSESSIONID=8FEB0A58F1E3E
898E342E07ADA12714A

User is maintaining session with server

Establishes session with same session identifier

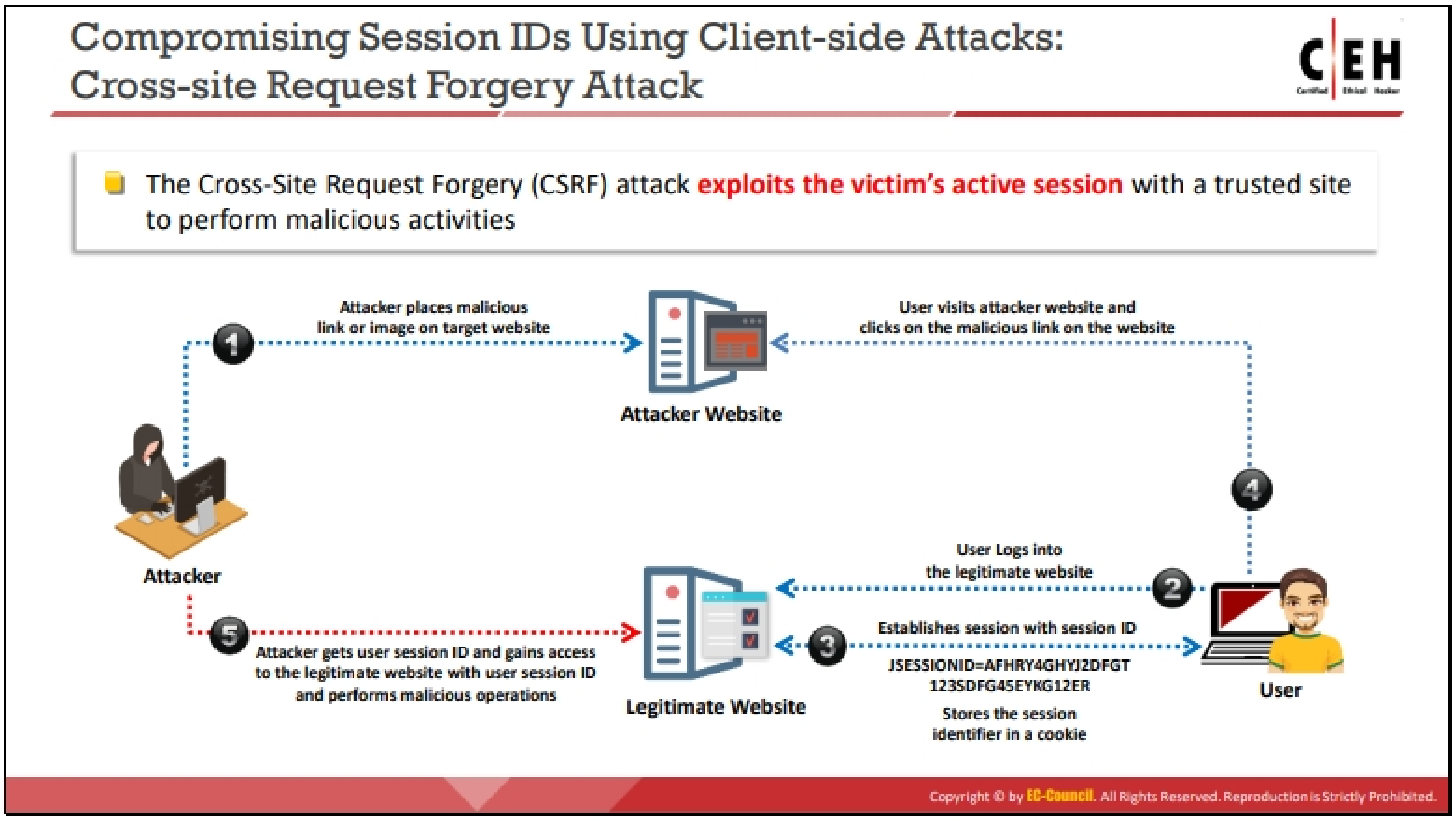
```
http://janaina:8080
http://janaina:8080 <SCRIPT>
alert(document.cookie);</SCRIPT>
```

Attacker sends malicious link with malicious JavaScript crafted in it

User clicks on that link; the malicious JavaScript runs

User follows the instructions made by the attacker and confirms the session identifier, thus attacker steals the session identifier

Figure 11.13: Prediction of session ID using a cross-site script attack



Compromising Session IDs Using Client-side Attacks: Cross-site Request Forgery Attack

- The Cross-Site Request Forgery (CSRF) attack **exploits the victim's active session** with a trusted site to perform malicious activities

Attacker
Attacker Website
Legitimate Website
User
1
Attacker places malicious link or image on target website
2
User Logs into the legitimate website
3
Establishes session with session ID
JSESSIONID=AFHRY4GHYJ2DFGT 123SDFG45EYKG12ER
Stores the session identifier in a cookie
4
User visits attacker website and clicks on the malicious link on the website
5
Attacker gets user session ID and gains access to the legitimate website with user session ID and performs malicious operations

Compromising Session IDs Using Client-side Attacks: Cross-site Request Forgery Attack

Cross-site request forgery (CSRF), also known as a one-click attack or session riding, is an attack in which the attacker exploits the victim's active session with a trusted site to perform malicious activities such as item purchases and the modification or retrieval of account information. In CSRF web attacks, the attacker creates a host form, containing malicious information, and sends it to the authorized user. The user fills in the form and sends it to the web server. Because the data originates from a trusted user, the web server accepts the data. Unlike an XSS attack, which exploits the trust a user has for a particular website, CSRF exploits the trust that a website has on a user's browser. A CSRF attack involves the following steps.

- The attacker hosts a web page with a form that appears legitimate. This page already contains the attacker's request.
- A user, believing the form to be the original, enters a login and password.
- Once the user completes the form, that page is submitted to the real site.
- The real site's server accepts the form, assuming that it was sent by the user based on the authentication credentials.

In this manner, the server accepts the attacker's request.

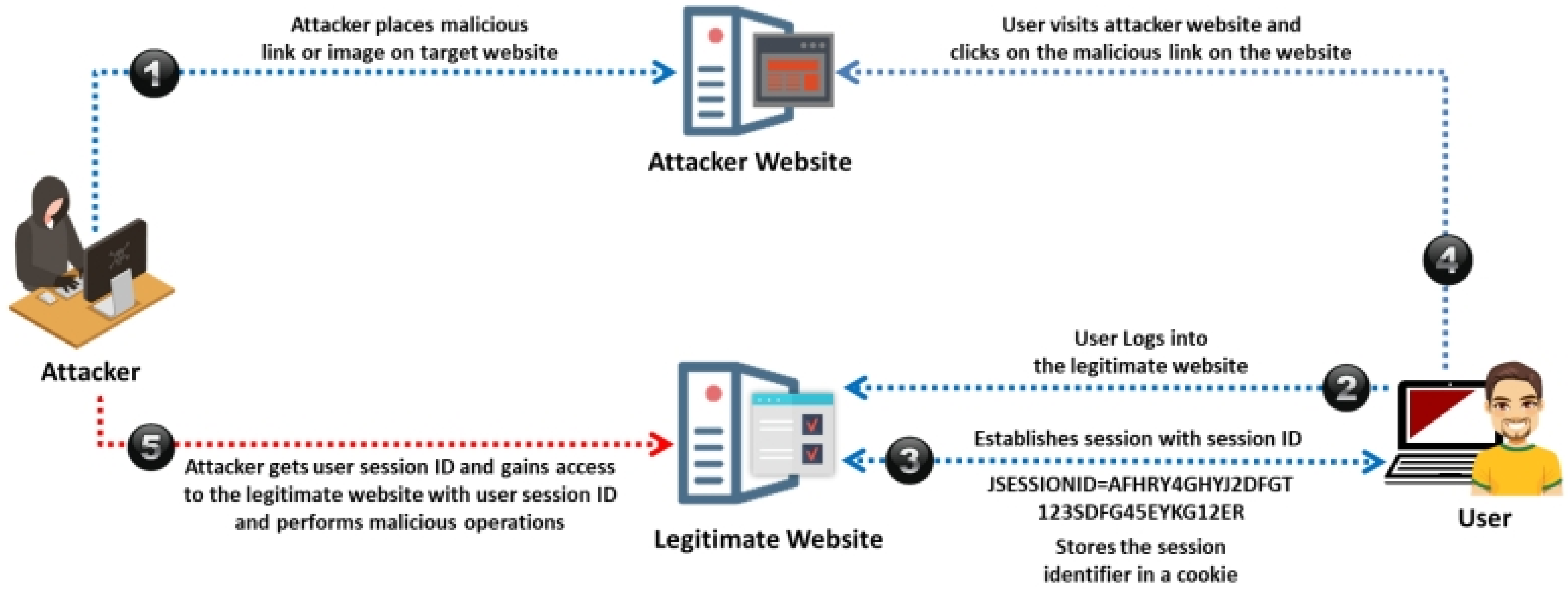
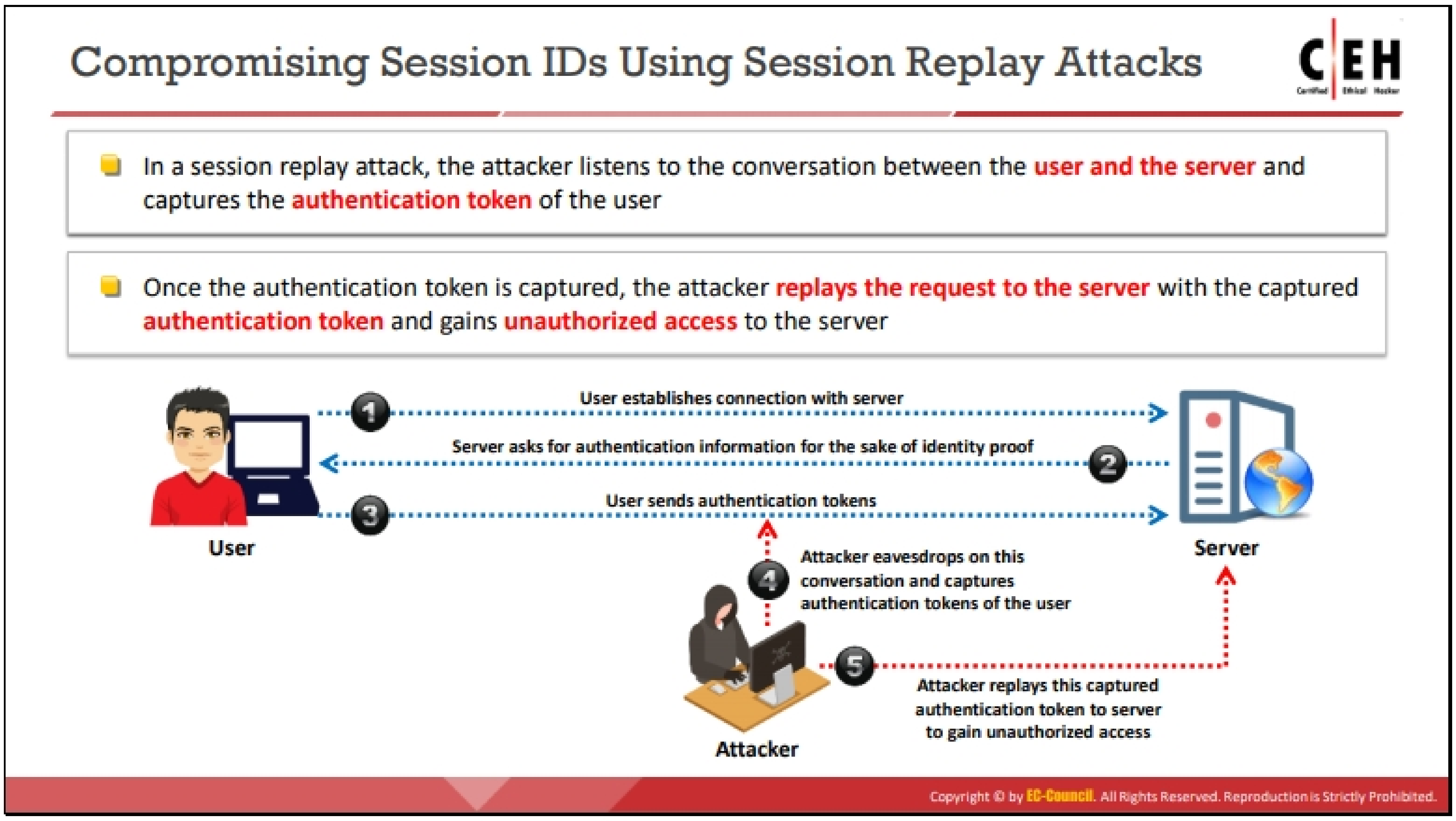

Attacker places malicious
link or image on target website
1
Attacker Website
User visits attacker website and
clicks on the malicious link on the website
4
Attacker
User Logs into
the legitimate website
2
Establishes session with session ID
3
JSESSIONID=AFHRY4GHYJ2DFGT
123SDFG45EYKG12ER
Stores the session
identifier in a cookie
User
5
Attacker gets user session ID and gains access
to the legitimate website with user session ID
and performs malicious operations
Legitimate Website

Figure 11.14: Prediction of session ID using a cross-site request forgery attack



Compromising Session IDs Using Session Replay Attacks

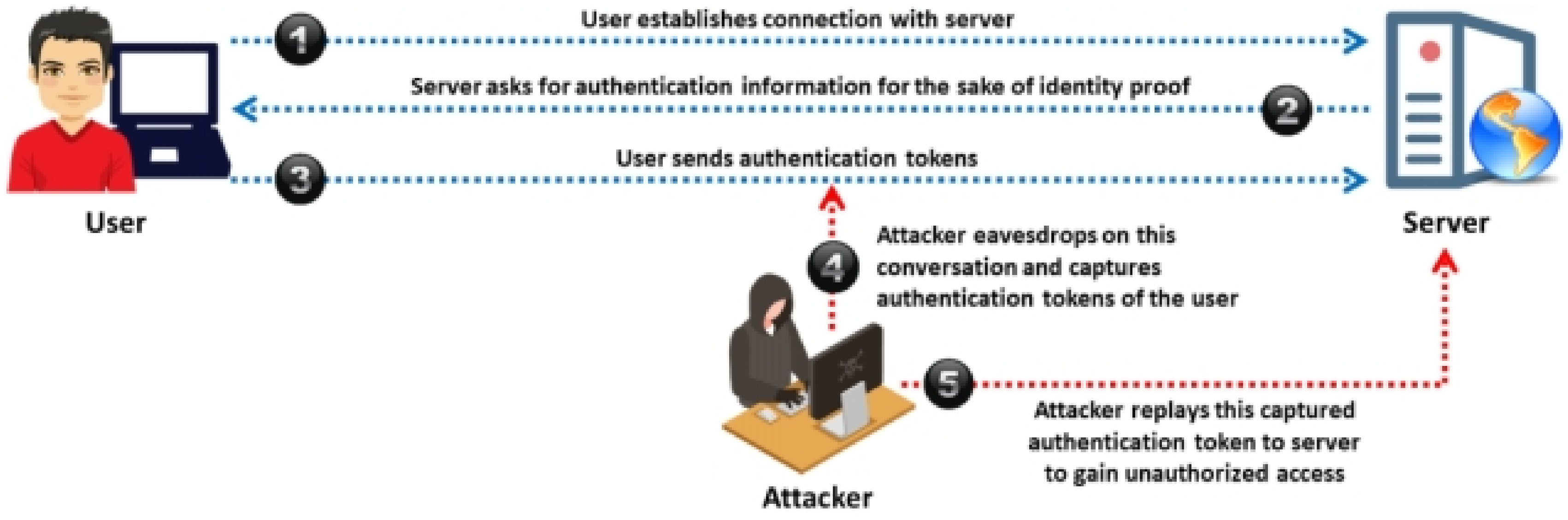
- In a session replay attack, the attacker listens to the conversation between the **user and the server** and captures the **authentication token** of the user
- Once the authentication token is captured, the attacker **replays the request to the server** with the captured **authentication token** and gains **unauthorized access** to the server

C|EH
1
User establishes connection with server
2
Server asks for authentication information for the sake of identity proof
3
User sends authentication tokens
User
Server
4
Attacker eavesdrops on this conversation and captures authentication tokens of the user
5
Attacker replays this captured authentication token to server to gain unauthorized access
Attacker

Compromising Session IDs Using Session Replay Attacks

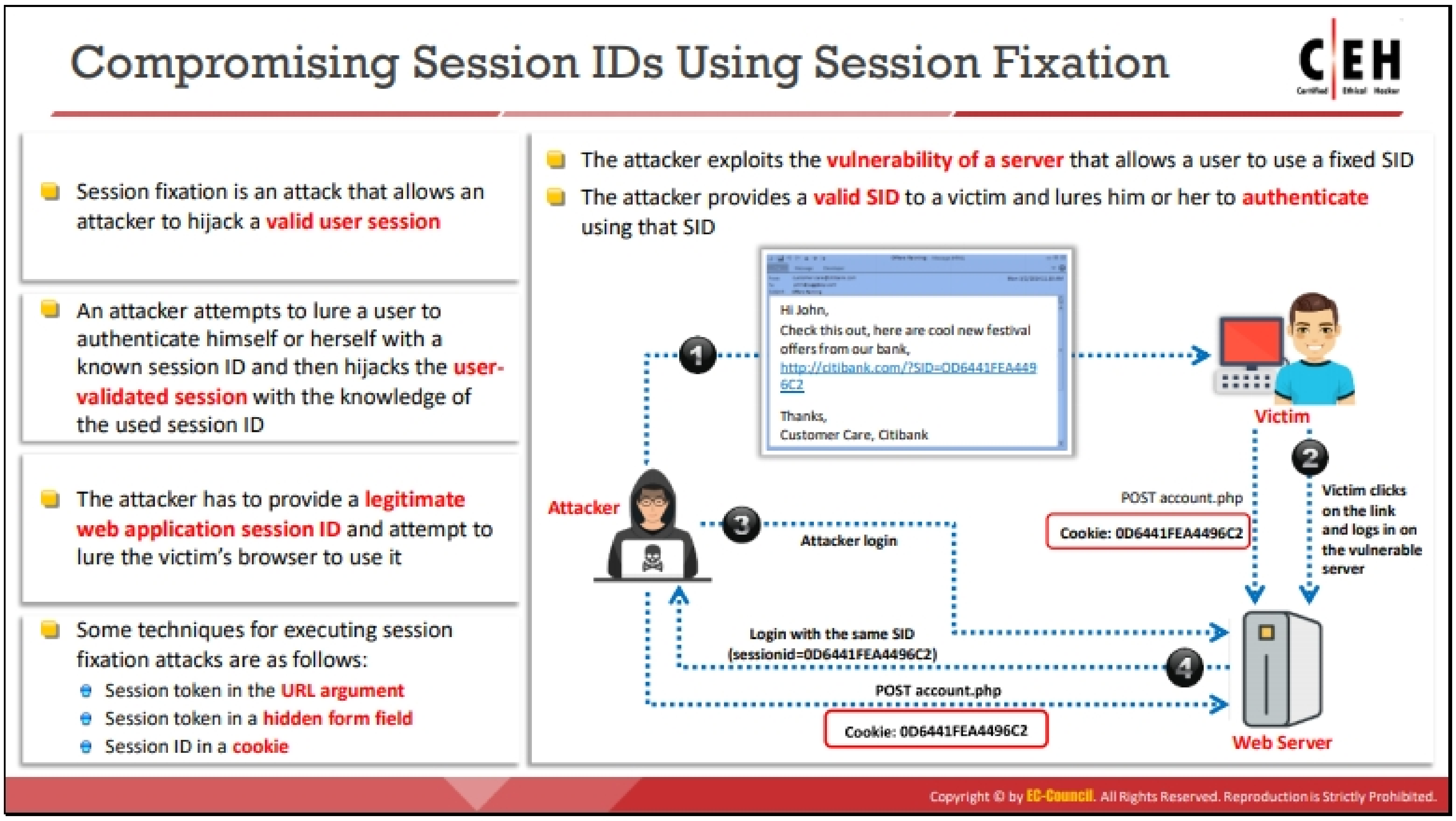
In a session replay attack, the attacker captures the authentication token of a user by listening to a conversation between the user and server. Once the authentication token is captured, the attacker replays the authentication request to the server with the captured authentication token to dodge the server; consequently, they gain unauthorized access to the server. A session replay attack involves the following steps.

- The user establishes a connection with the web server.
- The server asks the user for authentication information as identity proof.
- The user sends authentication tokens to the server. In this step, an attacker captures the authentication token of the user by eavesdropping on the conversation between the user and server.
- Once the authentication token is captured, the attacker replays the request to the server with the captured authentication token and gains unauthorized access to the server.



1
User establishes connection with server
2
Server asks for authentication information for the sake of identity proof
3
User sends authentication tokens
User
Server
4
Attacker eavesdrops on this conversation and captures authentication tokens of the user
5
Attacker replays this captured authentication token to server to gain unauthorized access
Attacker

Figure 11.15: Prediction of session ID using a session replay attack



Compromising Session IDs Using Session Fixation

- Session fixation is an attack that allows an attacker to hijack a **valid user session**
- An attacker attempts to lure a user to authenticate himself or herself with a known session ID and then hijacks the **user-validated session** with the knowledge of the used session ID
- The attacker has to provide a **legitimate web application session ID** and attempt to lure the victim's browser to use it
- Some techniques for executing session fixation attacks are as follows:
  - Session token in the **URL argument**
  - Session token in a **hidden form field**
  - Session ID in a **cookie**

- The attacker exploits the **vulnerability of a server** that allows a user to use a fixed SID
- The attacker provides a **valid SID** to a victim and lures him or her to **authenticate** using that SID

Hi John,
Check this out, here are cool new festival offers from our bank,
http://citibank.com/?SID=0D6441FEA4496C2

Thanks,
Customer Care, Citibank

Compromising Session IDs Using Session Fixation

Web session security prevents an attacker from intercepting, brute forcing, or predicting the session ID issued by a web server to a user's browser as proof of an authenticated session. However, this approach ignores the possibility of the attacker issuing a session ID to the user's browser, forcing it to use the chosen session ID. This type of attack is called a session fixation attack because an attacker fixes the user's session ID in advance, instead of generating it randomly at the time of login.

The attacker performs a session fixation attack to hijack a valid user session. The attacker takes advantage of limitations in web-application session ID management. Web applications allow the user to authenticate themselves using an existing session ID, instead of generating a new session ID. In this type of attack, the attacker provides a legitimate web-application session ID and lures the victim to use it. If the victim's browser uses that session ID, then the attacker can hijack the user-validated session because the attacker is already aware of the session ID used by the victim.

A session fixation attack is a kind of session hijack. However, instead of stealing the session established between a user and web server after the user logs in, a session fixation attack fixes an established session on the user's browser; thus, the attack is initiated before the user logs in. An attacker uses various techniques to perform a session fixation attack:

- Session token in the URL argument
- Session token in a hidden form field
- Session ID in a cookie

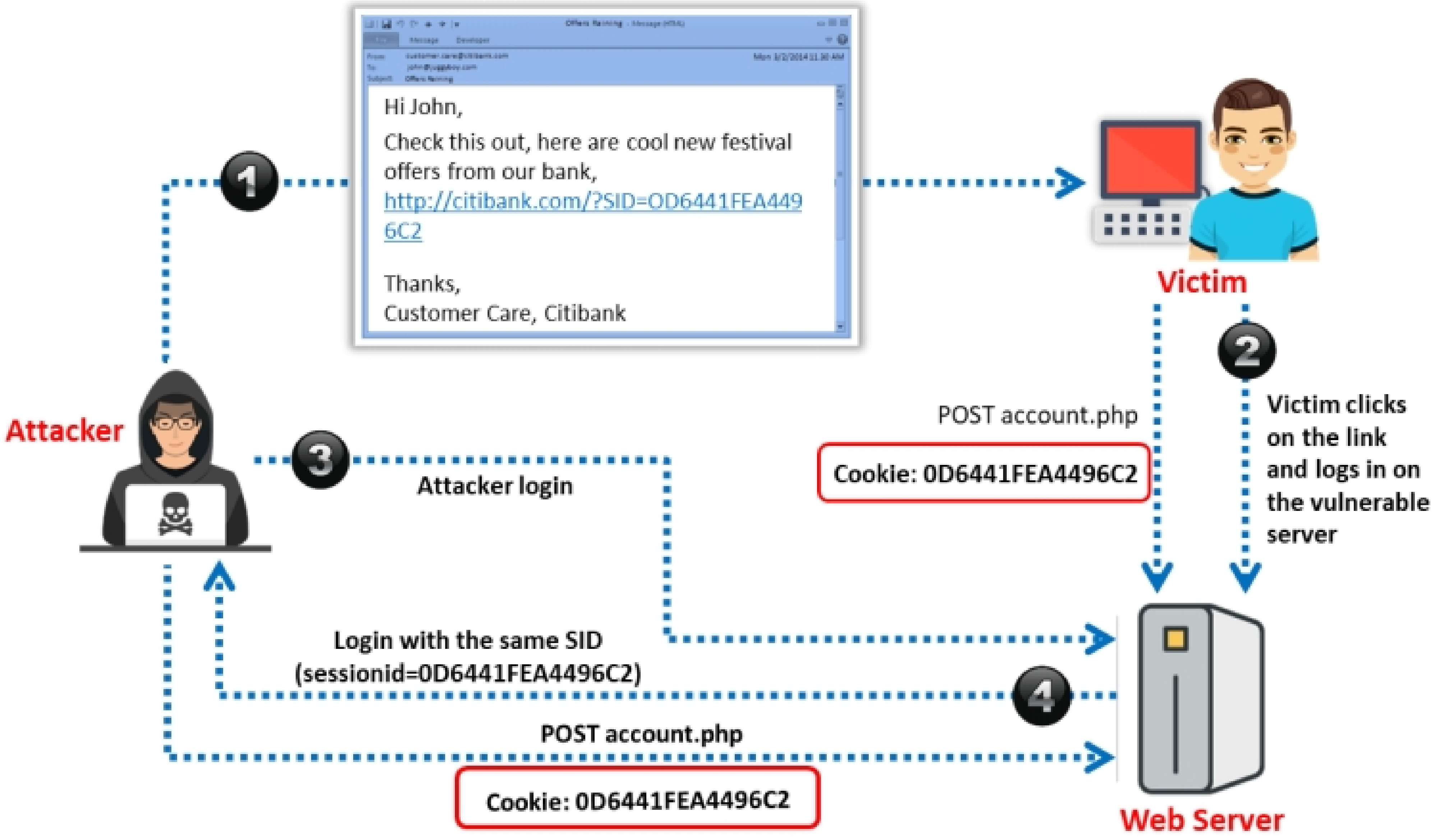
The attacker must choose a technique based on how the target web application uses session tokens. The attacker exploits the vulnerability of a server that allows a user to use a fixed session ID. Then, the attacker provides a valid session ID to a victim and lures him to authenticate themselves using that session ID. A session fixation attack has the following three phases.

- **Session set-up phase**: In this phase, the attacker first obtains a legitimate session ID by establishing a connection with the target web server. Few web servers support the idle session time-out feature. If the target web server supports this feature, the attacker needs to send requests repeatedly to keep the established trap session ID alive.
- **Fixation phase**: In this phase, the attacker introduces the session ID to the victim's browser, thereby fixing the session.
- **Entrance phase**: In this phase, the attacker waits for the victim to log in to the target web server using the trap session ID and then enters the victim's session.

A session fixation attack is performed through the following steps.

- First, the attacker establishes a legitimate connection with the target web server.
- The target web server (e.g., *http://citibank.com/*) issues a session ID, say 0D6441FEA4496C2, to the attacker.
- The attacker sends a link with the established session ID, say *http://citibank.com/?SID=0D6441FEA4496C2*, to the victim and lures the victim to click on it to access the website.
- The victim clicks on the link, believing it to be a legitimate link sent by the bank. This opens the server's login page in the victim's browser for SID=0D6441FEA4496C2.
- The web server checks that the session ID 0D6441FEA4496C2 is already established and is in an active state; hence, it does not create the new session.
- Finally, the victim enters their login credentials in the login script, and the server grants them access to the bank account.
- At this point, knowing the session ID, the attacker can also access the victim's bank account via *http://citibank.com/? SID=0D6441FEA4496C2*.

Because the session ID is set by the attacker before the user logged in, the user can be said to have logged into the attacker's session.



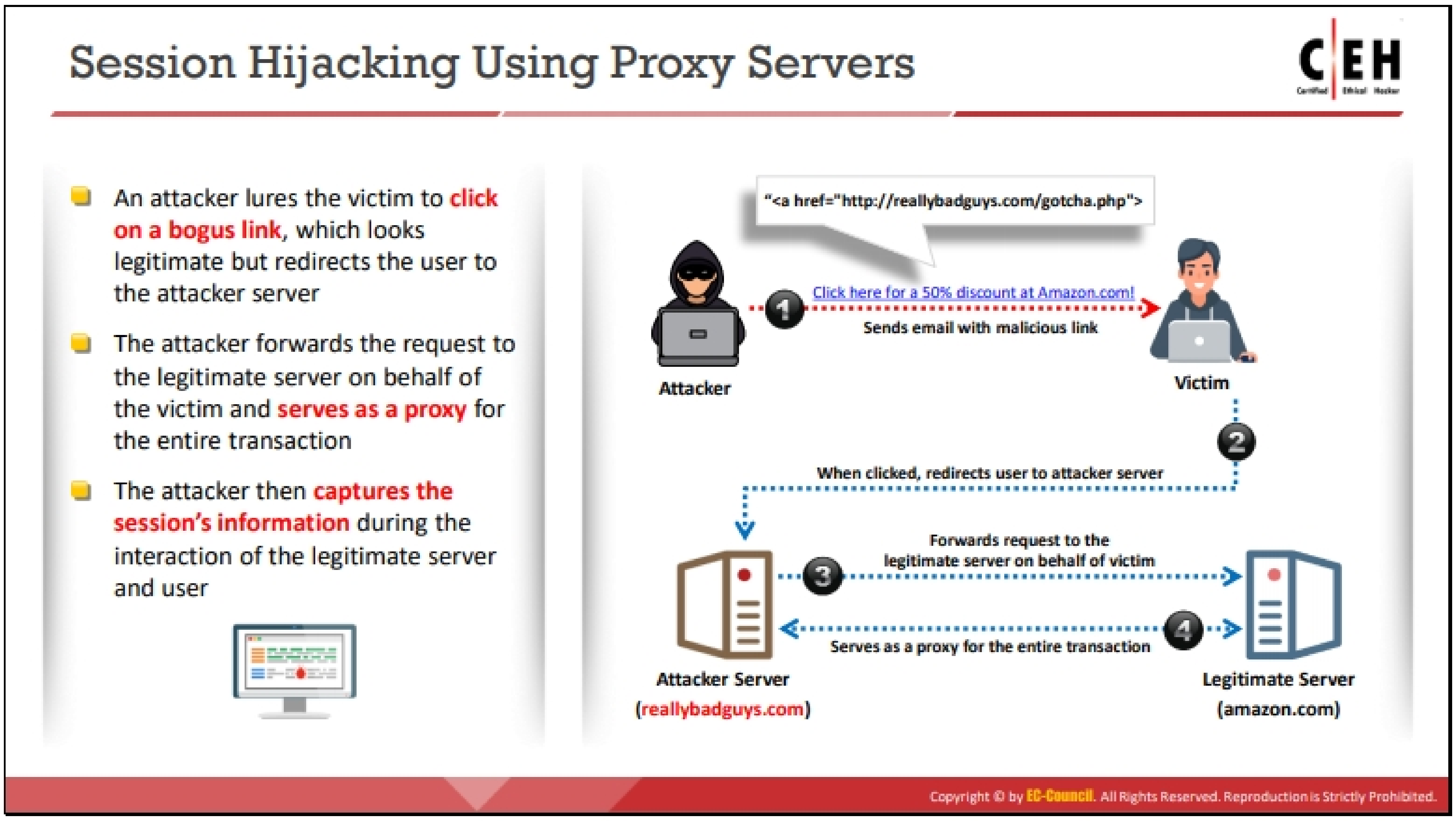
Hi John,

Check this out, here are cool new festival offers from our bank,
http://citibank.com/?SID=OD6441FEA4496C2

Thanks,
Customer Care, Citibank

1
Victim
2
POST account.php
Cookie: 0D6441FEA4496C2
Victim clicks on the link and logs in on the vulnerable server
Attacker
3
Attacker login
Login with the same SID (sessionid=0D6441FEA4496C2)
4
POST account.php
Cookie: 0D6441FEA4496C2
Web Server

Figure 11.16: Session fixation attack



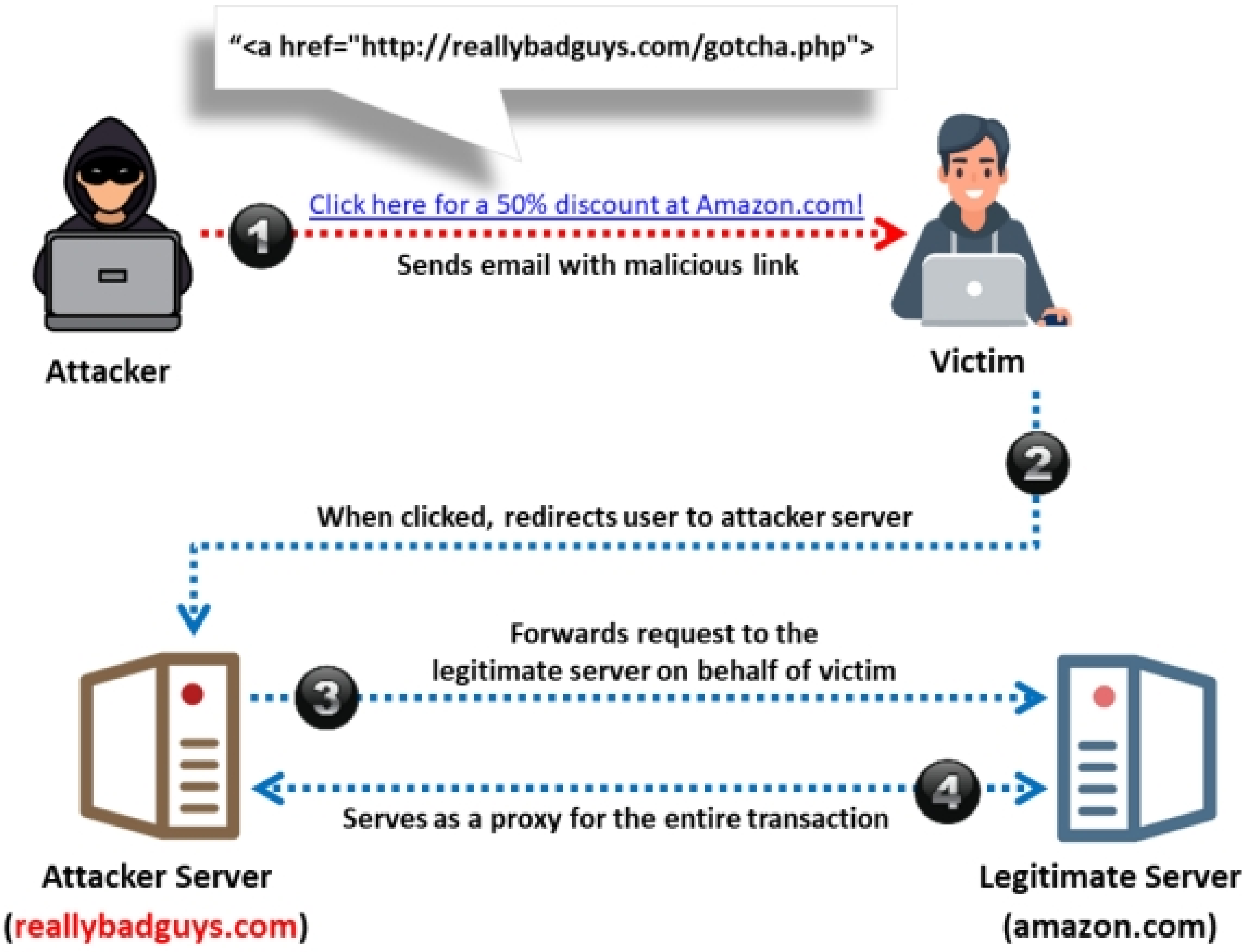
Session Hijacking Using Proxy Servers

- An attacker lures the victim to **click on a bogus link**, which looks legitimate but redirects the user to the attacker server
- The attacker forwards the request to the legitimate server on behalf of the victim and **serves as a proxy** for the entire transaction
- The attacker then **captures the session's information** during the interaction of the legitimate server and user

"<a href="http://reallybadguys.com/gotcha.php">
Attacker
1
Click here for a 50% discount at Amazon.com!
Sends email with malicious link
Victim
2
When clicked, redirects user to attacker server
3
Forwards request to the legitimate server on behalf of victim
4
Serves as a proxy for the entire transaction
Attacker Server
(reallybadguys.com)
Legitimate Server
(amazon.com)

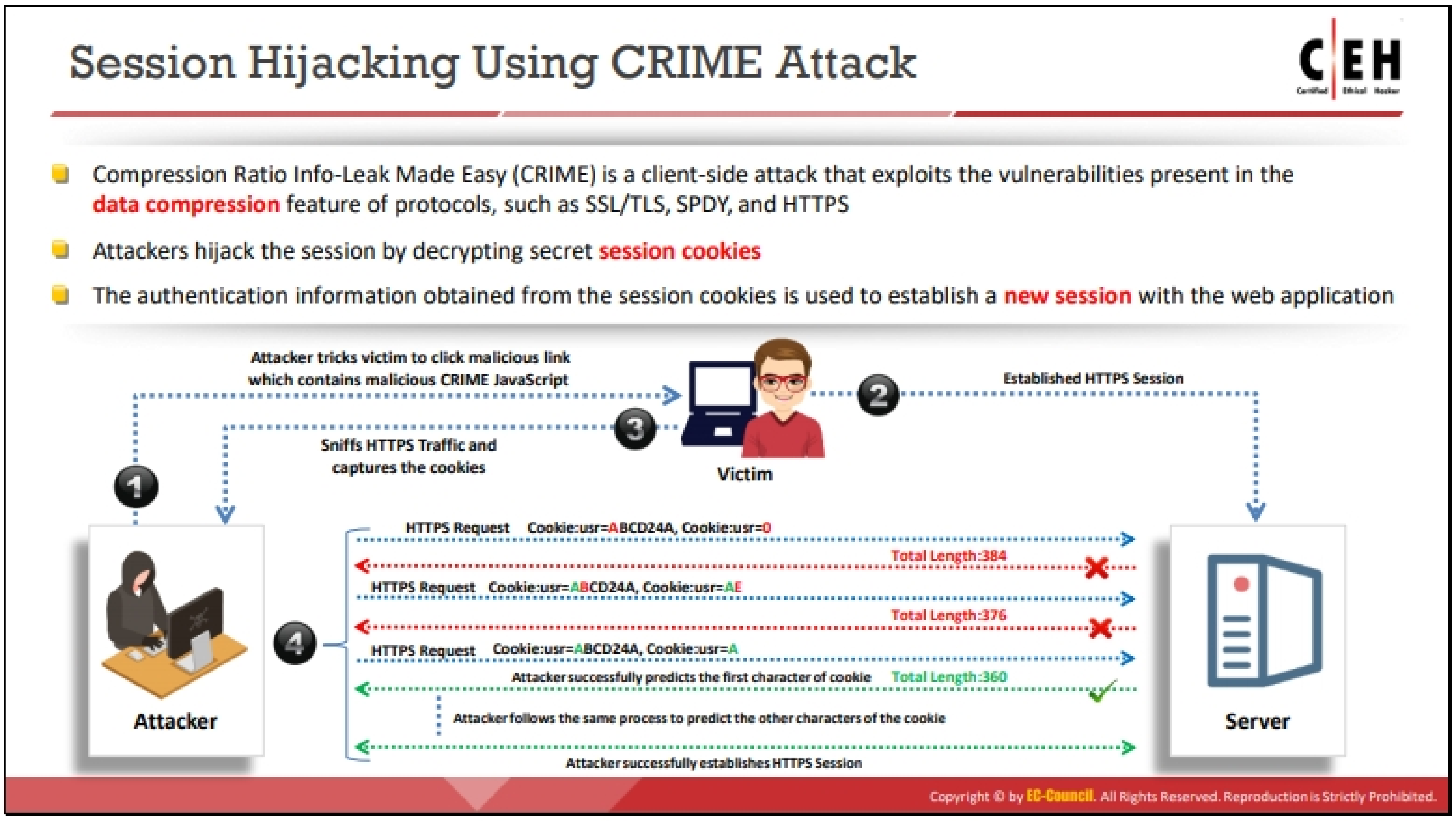
Session Hijacking Using Proxy Servers

An attacker lures the victim to click on a fake link, which appears legitimate but redirects the user to the attacker's server. The attacker then forwards the request to the legitimate server on behalf of the victim and serves as a proxy for the entire transaction. Acting as a proxy, the attacker captures the session information during the interaction between the legitimate server and user.



"<a href="http://reallybadguys.com/gotcha.php">
Attacker
1
Click here for a 50% discount at Amazon.com!
Sends email with malicious link
Victim
2
When clicked, redirects user to attacker server
3
Forwards request to the legitimate server on behalf of victim
4
Serves as a proxy for the entire transaction
Attacker Server
(reallybadguys.com)
Legitimate Server
(amazon.com)

Figure 11.17: Session hijacking using proxy servers



Session Hijacking Using CRIME Attack

- Compression Ratio Info-Leak Made Easy (CRIME) is a client-side attack that exploits the vulnerabilities present in the **data compression** feature of protocols, such as SSL/TLS, SPDY, and HTTPS
- Attackers hijack the session by decrypting secret **session cookies**
- The authentication information obtained from the session cookies is used to establish a **new session** with the web application

Attacker tricks victim to click malicious link which contains malicious CRIME JavaScript
Established HTTPS Session
Sniffs HTTPS Traffic and captures the cookies
Victim
1
2
3
4
Attacker
Server
HTTPS Request Cookie:usr=ABCD24A, Cookie:usr=0
Total Length:384
HTTPS Request Cookie:usr=ABCD24A, Cookie:usr=AE
Total Length:376
HTTPS Request Cookie:usr=ABCD24A, Cookie:usr=A
Attacker successfully predicts the first character of cookie
Total Length:360
Attacker follows the same process to predict the other characters of the cookie
Attacker successfully establishes HTTPS Session

Session Hijacking Using CRIME Attack

Compression Ratio Info-Leak Made Easy (CRIME) is a client-side attack that exploits vulnerabilities in the data-compression feature of protocols such as SSL/Transport Layer Security (TLS), SPDY, and HTTP Secure (HTTPS). The possibility of mitigation against HTTPS compression is low, which makes this vulnerability even more dangerous than other compression vulnerabilities.

When two hosts on the Internet establish a connection using HTTPS, a TLS session is established, and the data are transmitted in an encrypted form. Hence, it is difficult for an attacker to read or modify the messages between the two hosts. When a user logs into a web application, authentication data are stored in a cookie. Whenever the browser sends an HTTPS request to the web application, the stored cookie is used for authentication. In this attack, the attacker attempts to access the authentication cookie to hijack the victim's session.

In HTTPS, cookies are compressed using a lossless data compression algorithm (DEFLATE) and then encrypted. Hence, it is difficult for an attacker to obtain the value of the cookie with simple sniffing.

To perform a CRIME attack, an attacker must use social engineering techniques to trick the victim into clicking on a malicious link. When the victim clicks on the malicious link, it either injects malicious code into the victim's system or redirects the victim to a malicious website. If the victim has already established an HTTPS connection with a secured web application, the attacker sniffs the victim's HTTPS traffic using techniques such as ARP spoofing. Through sniffing, the attacker captures the cookie value from the HTTPS messages and sends multiple HTTPS requests to the web application with that cookie prepended with a few random characters. Subsequently, the attacker monitors the traffic between the victim and web application to obtain the compressed and encrypted value of the cookie. After capturing the cookie, the attacker analyzes the cookie length and predicts the actual value of the authentication cookie.

After obtaining the authentication cookie, the attacker impersonates the victim and hijacks the victim's session with the secure web application to steal confidential information such as passwords, social security numbers, and credit card numbers. Attackers use tools such as CrimeCheck to detect whether a web server has TLS or HTTP compression enabled and are thus vulnerable to CRIME attacks.

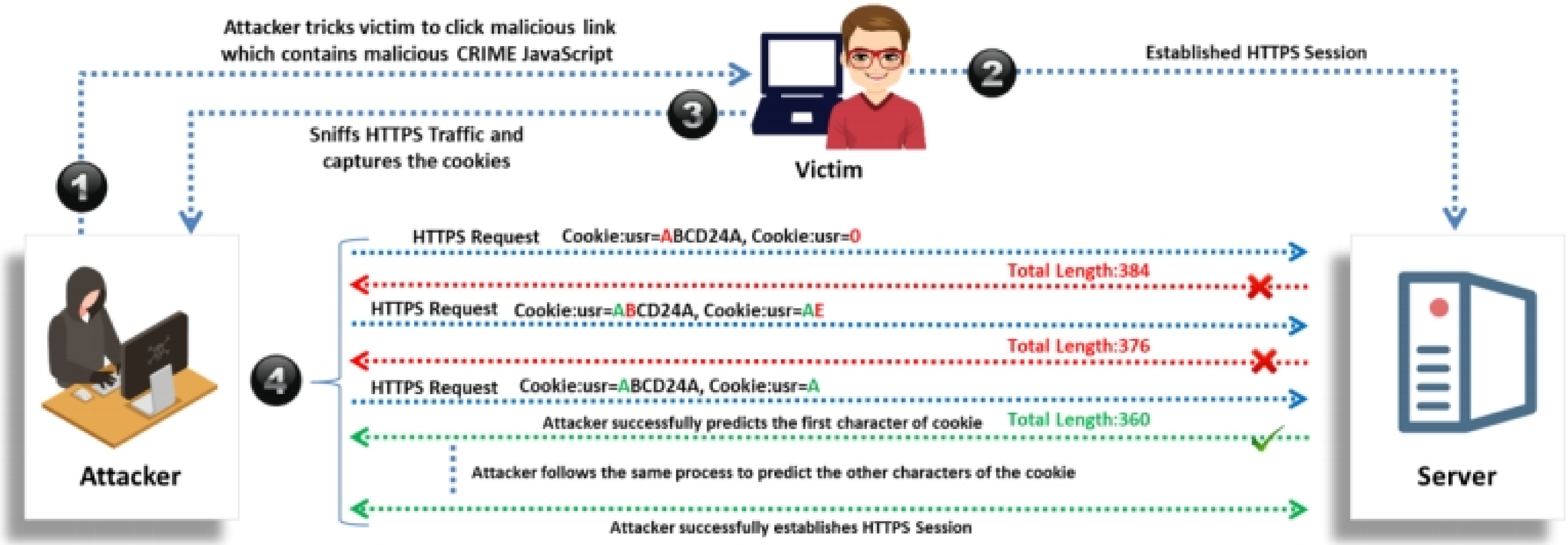
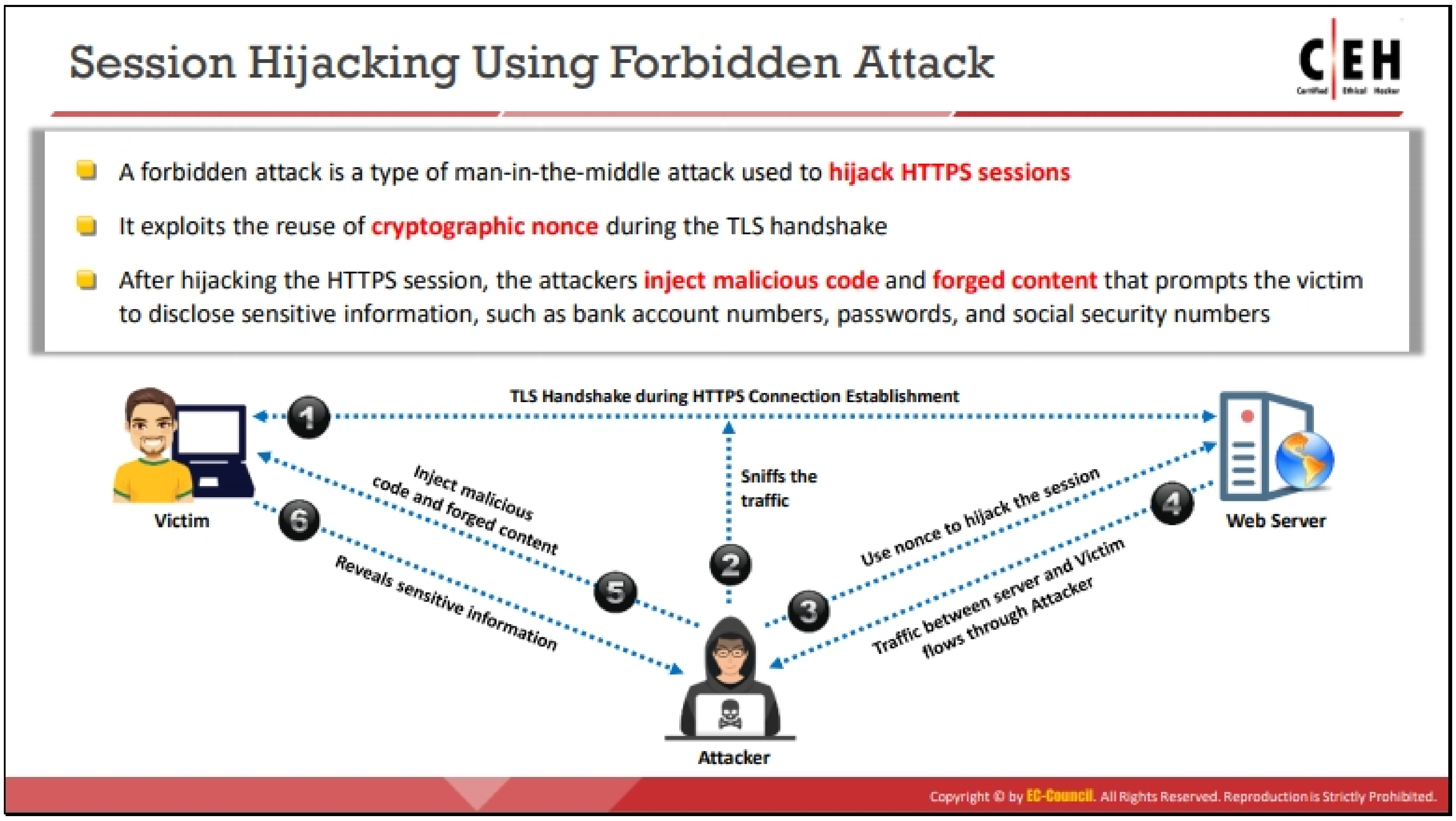

Attacker tricks victim to click malicious link which contains malicious CRIME JavaScript
1
2
Established HTTPS Session
3
Sniffs HTTPS Traffic and captures the cookies
Victim
Attacker
4
HTTPS Request Cookie:usr=ABCD24A, Cookie:usr=0
Total Length:384
HTTPS Request Cookie:usr=ABCD24A, Cookie:usr=AE
Total Length:376
HTTPS Request Cookie:usr=ABCD24A, Cookie:usr=A
Attacker successfully predicts the first character of cookie
Total Length:360
Attacker follows the same process to predict the other characters of the cookie
Attacker successfully establishes HTTPS Session
Server

Figure 11.18: Session hijacking using a CRIME attack



Session Hijacking Using Forbidden Attack

- A forbidden attack is a type of man-in-the-middle attack used to **hijack HTTPS sessions**
- It exploits the reuse of **cryptographic nonce** during the TLS handshake
- After hijacking the HTTPS session, the attackers **inject malicious code** and **forged content** that prompts the victim to disclose sensitive information, such as bank account numbers, passwords, and social security numbers

TLS Handshake during HTTPS Connection Establishment
Victim
Web Server
Attacker
1
2
Sniffs the traffic
3
Use nonce to hijack the session
4
Traffic between server and Victim flows through Attacker
5
Inject malicious code and forged content
6
Reveals sensitive information

Session Hijacking Using Forbidden Attack

A forbidden attack is a type of MITM attack that can be executed when a cryptographic nonce is reused while establishing an HTTPS session with a server. According to the TLS specification, these arbitrary pieces of data must be used once. This attack exploits the vulnerability that the TLS implementation incorrectly reuses the same nonce when data are encrypted using the Advanced Encryption Standard–Galois/Counter Mode (AES-GCM) during the TLS handshake. Attackers exploit this vulnerability to perform an MITM attack by generating cryptographic keys used for authentication. Repeating the same nonce during the TLS handshake allows an attacker to monitor and hijack the connection. After hijacking the HTTPS session and bypassing the protection, attackers inject malicious code and forged content into the transmission, such as JavaScript code or web fields that prompt the user to disclose passwords, social security numbers, or other confidential information. A forbidden attack involves the following steps.

- The attacker monitors the connection between the victim and web server and sniffs the nonce from the TLS handshake messages.
- The attacker generates authentication keys using the nonce and hijacks the connection.
- All the traffic between the victim and web server flows through the attacker's machine.
- The attacker injects JavaScript code or web fields into the transmission towards the victim.
- The victim reveals sensitive information such as bank account numbers, passwords, and social security numbers to the attacker.

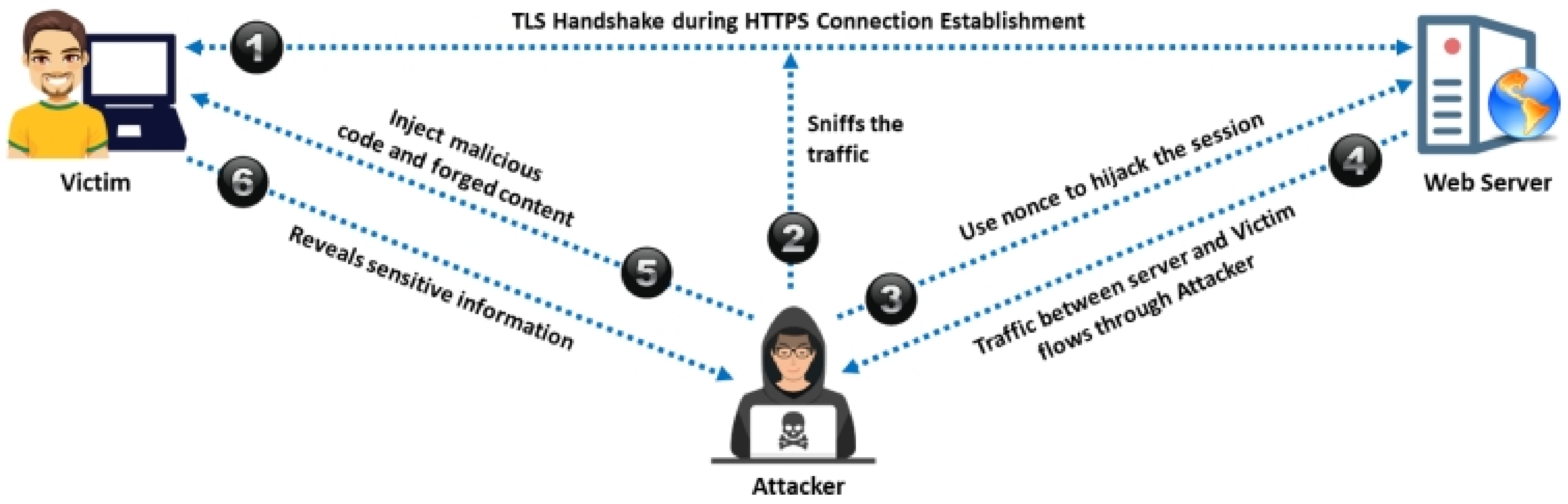
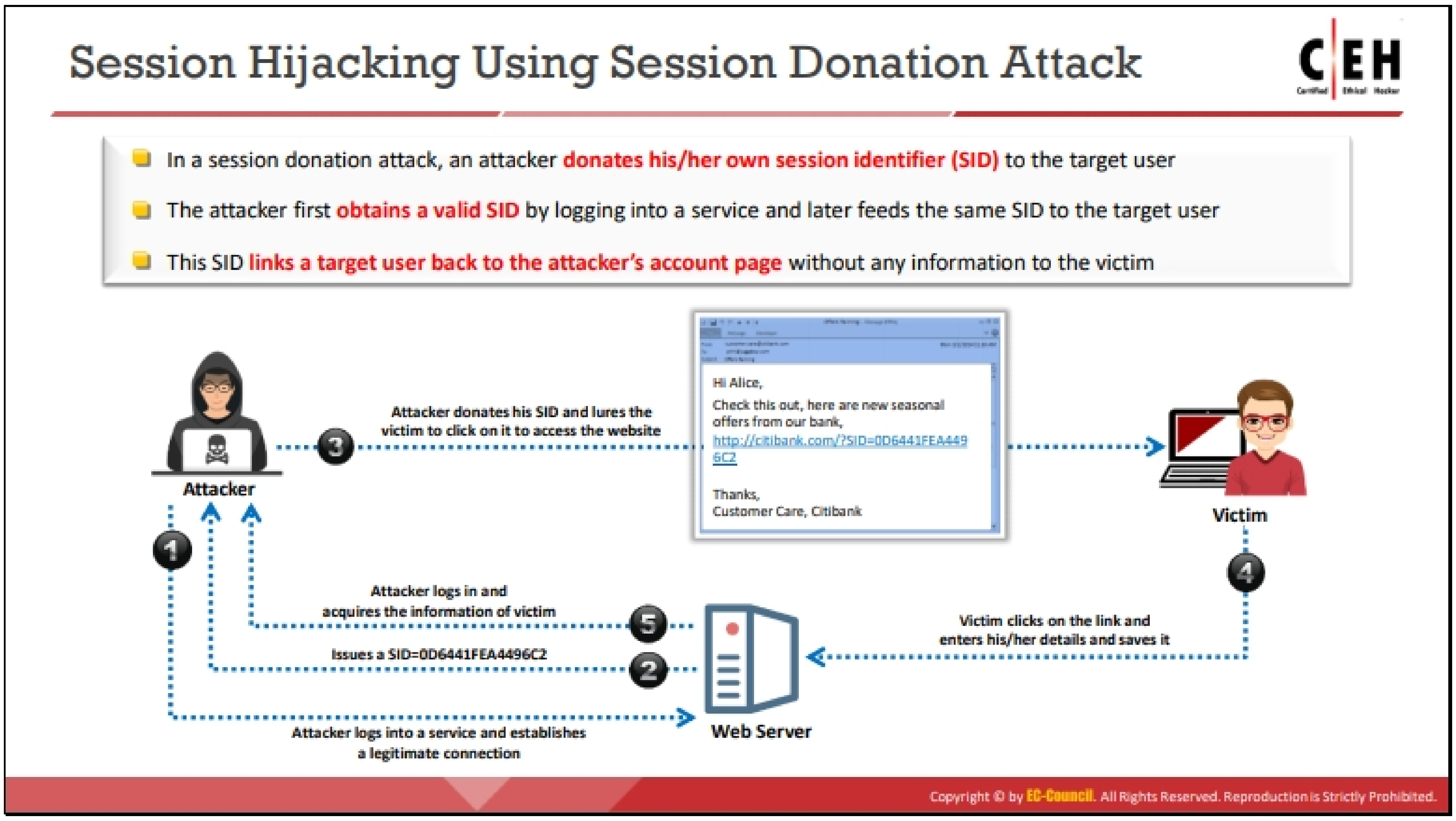

TLS Handshake during HTTPS Connection Establishment
1
Victim
Web Server
Sniffs the traffic
2
Inject malicious code and forged content
5
Use nonce to hijack the session
3
Traffic between server and Victim flows through Attacker
4
Reveals sensitive information
6
Attacker

Figure 11.19: Session hijacking using a forbidden attack



Session Hijacking Using Session Donation Attack

- In a session donation attack, an attacker **donates his/her own session identifier (SID)** to the target user
- The attacker first **obtains a valid SID** by logging into a service and later feeds the same SID to the target user
- This SID **links a target user back to the attacker's account page** without any information to the victim

Hi Alice,

Check this out, here are new seasonal offers from our bank,
http://citibank.com/?SID=0D6441FEA4496C2

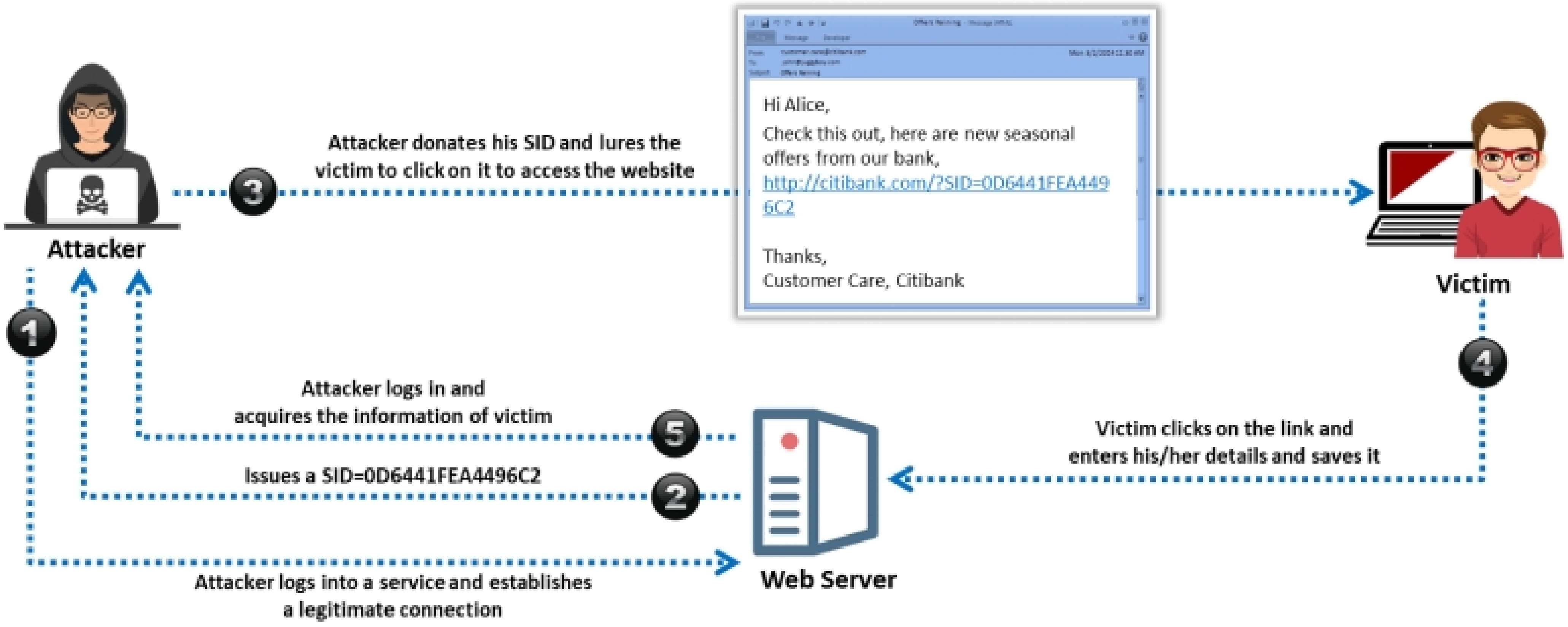
Thanks,
Customer Care, Citibank

Attacker
Attacker donates his SID and lures the victim to click on it to access the website
Victim
Attacker logs in and acquires the information of victim
Issues a SID=0D6441FEA4496C2
Victim clicks on the link and enters his/her details and saves it
Attacker logs into a service and establishes a legitimate connection
Web Server
1
2
3
4
5

Session Hijacking Using Session Donation Attack

In a session donation attack, the attacker donates their own session ID to the target user. In this attack, the attacker first obtains a valid session ID by logging into a service and later feeds the same session ID to the target user. This session ID links a target user to the attacker's account page without disclosing any information to the victim. When the target user clicks on the link and enters the details (username, password, payment details, etc.) in a form, the entered details are linked to the attacker's account. To initiate this attack, the attacker can send their session ID using techniques such as cross-site cooking, an MITM attack, and session fixation. A session donation attack involves the following steps.

- First, the attacker logs into a service, establishes a legitimate connection with the target web server, and deletes the stored information.
- The target web server (e.g., *http://citibank.com/*) issues a session ID, say 0D6441FEA4496C2, to the attacker.
- The attacker then donates their session ID, say *http://citibank.com/?SID=0D6441FEA4496C2*, to the victim and lures the victim to click on it to access the website.
- The victim clicks on the link, believing it to be a legitimate link sent by the bank. This opens the server's page in the victim's browser with SID=0D6441FEA4496C2. Finally, the victim enters their information in the page and saves it.
- The attacker can now login as themselves and acquire the victim's information.



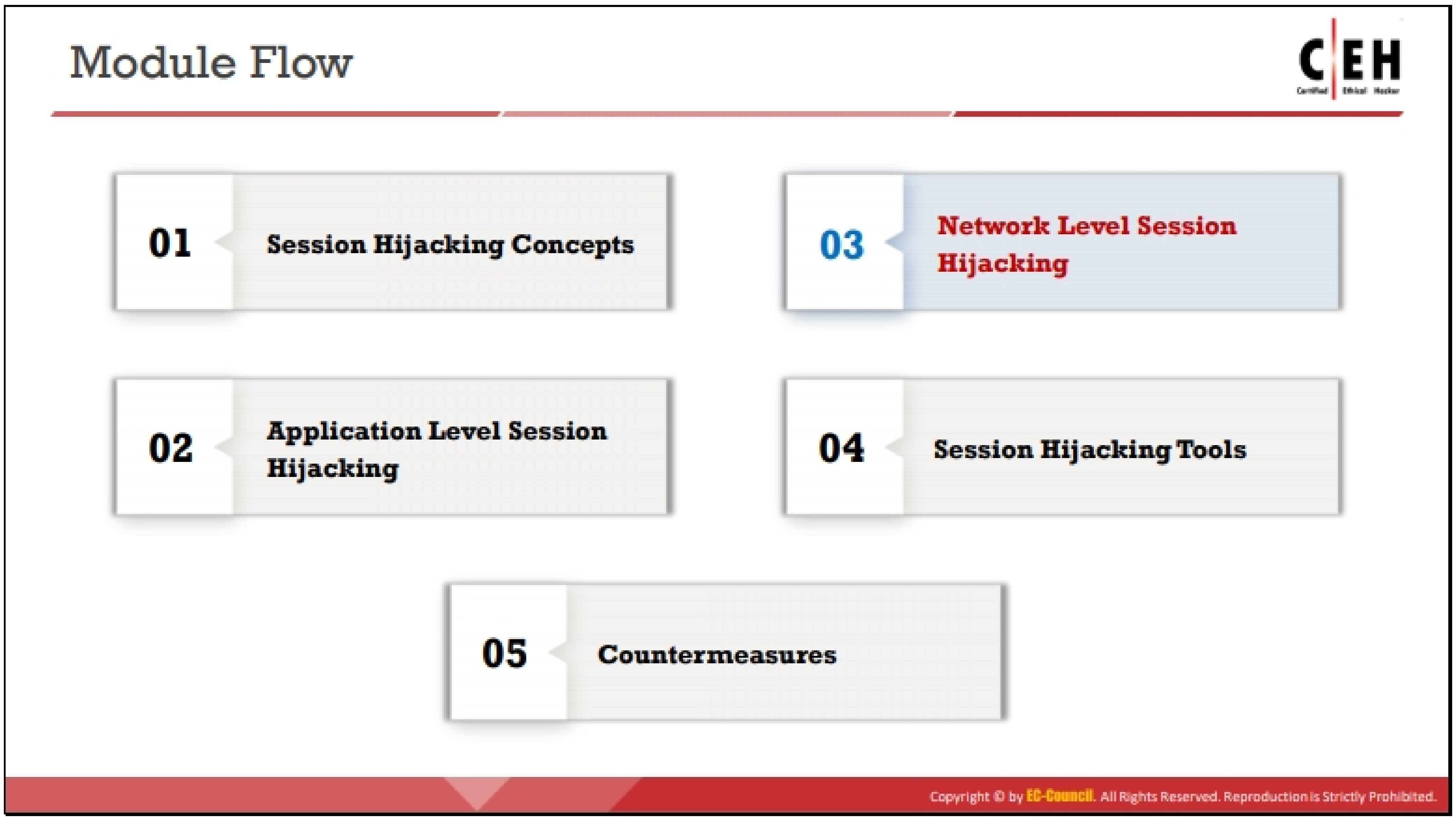
Hi Alice,

Check this out, here are new seasonal offers from our bank,
http://citibank.com/?SID=0D6441FEA4496C2

Thanks,
Customer Care, Citibank

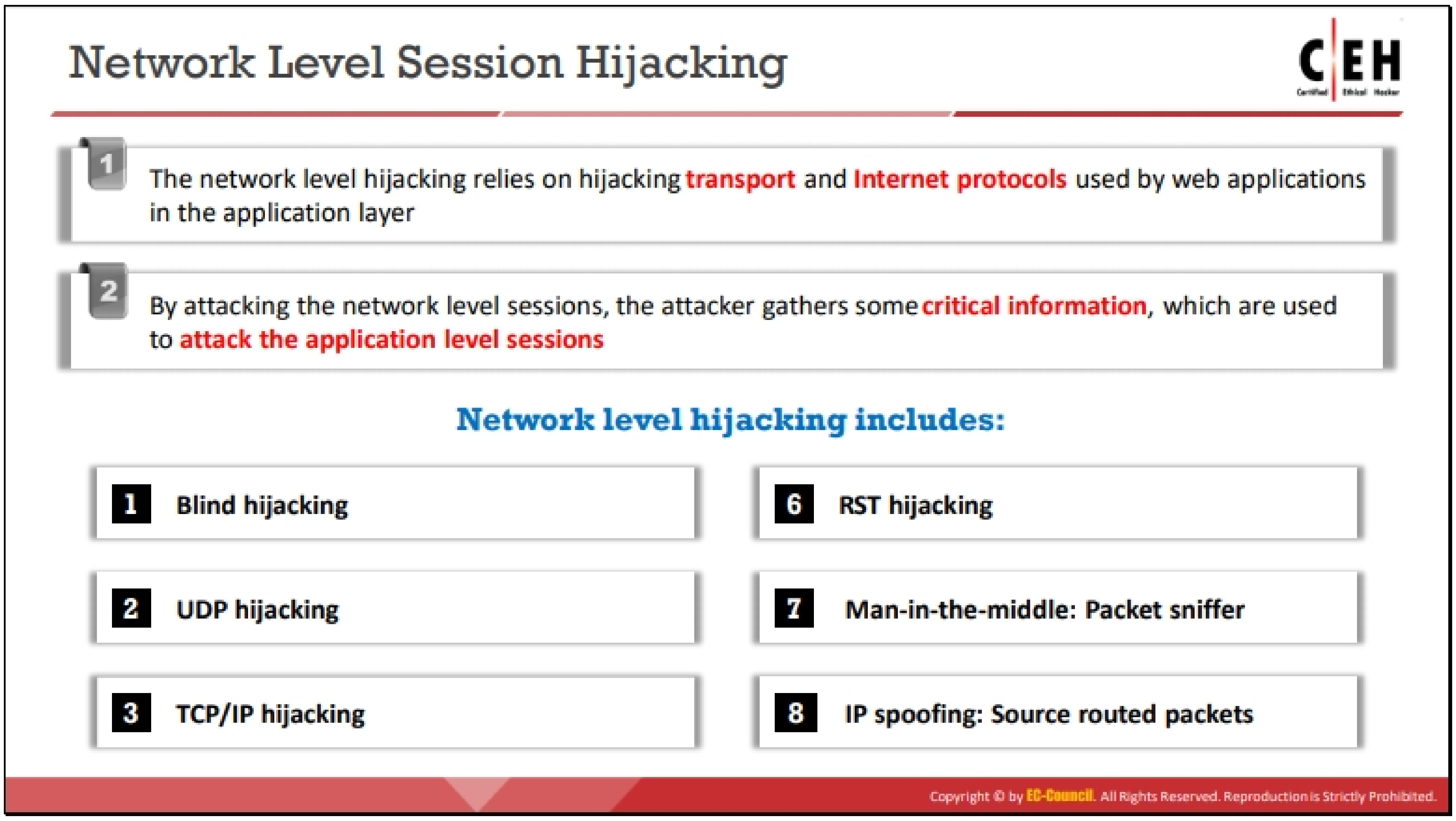
Attacker
Victim
Web Server
1
Attacker logs into a service and establishes a legitimate connection
2
Issues a SID=0D6441FEA4496C2
3
Attacker donates his SID and lures the victim to click on it to access the website
4
Victim clicks on the link and enters his/her details and saves it
5
Attacker logs in and acquires the information of victim

Figure 11.20: Session hijacking using a session donation attack



Module Flow

- **01** Session Hijacking Concepts
- **02** Application Level Session Hijacking
- **03** Network Level Session Hijacking
- **04** Session Hijacking Tools
- **05** Countermeasures



Network Level Session Hijacking

1. The network level hijacking relies on hijacking **transport** and **Internet protocols** used by web applications in the application layer
2. By attacking the network level sessions, the attacker gathers some **critical information**, which are used to **attack the application level sessions**

Network level hijacking includes:

1. Blind hijacking
2. UDP hijacking
3. TCP/IP hijacking
6. RST hijacking
7. Man-in-the-middle: Packet sniffer
8. IP spoofing: Source routed packets

Network Level Session Hijacking

Attackers especially focus on network level session hijacking because it does not require host access, in contrast to host-level session hijacking, or a need to tailor their attacks on a per-application basis, in contrast to application level hijacking.

This section discusses network level session hijacking, concepts related to network communications, and various techniques used to perform network level session hijacking.

Network level hijacking relies on hijacking transport and Internet protocols used by web applications in the application layer. By attacking network level sessions, the attacker gathers some critical information that is used to attack application level sessions.

The following are different types of network level hijacking:

- Blind hijacking
- UDP hijacking
- TCP/IP hijacking
- RST hijacking
- Man-in-the-middle: packet sniffer
- IP spoofing: source routed packets

Three-way Handshake

When two parties establish a connection using TCP, they perform a three-way handshake. A three-way handshake starts the connection and exchanges all the parameters needed for the two parties to communicate. TCP uses a three-way handshake to establish a new connection.

Initially, the client-side connection is in the closed state and the server-side in the listening state. The client initiates the connection by sending the initial sequence number (ISN) and setting the SYN flag. The client is now in the SYN-SENT state.

When the server receives this packet, it acknowledges the client sequence number and sends its own ISN with the SYN flag set. The server's state is now SYN-RECEIVED. On receipt of this packet, the client acknowledges the server sequence number by incrementing it and setting the ACK flag. The client is now in the established state. At this point, the two machines have established a session and can communicate.

On receiving the client's acknowledgement, the server enters the established state and sends an acknowledgment, incrementing the client's sequence number. The connection can be closed either by using the FIN or RST flag or through a time out.

If the RST flag of a packet is set, the receiving host enters the CLOSED state and frees all resources associated with this connection. This leads to the connection drop of any additional incoming packets.

If the packet is sent with the FIN flag turned on, the receiving host closes the connection because it enters the CLOSE-WAIT state. The packets sent by the client are accepted in an established connection if the sequence number is within the range and follows its predecessor.

If the sequence number is beyond the range of the acceptable sequence numbers, it drops the packet and sends an ACK packet using the expected sequence number.

For the three parties to communicate, the following information is required:

- IP address
- Port numbers
- Sequence numbers

It is easy for an attacker to determine the IP address and port number; these are available in the IP packets, which do not change throughout the session. However, the sequence numbers change. Therefore, the attacker must successfully guess the sequence numbers for a blind hijack. If the attacker can fool the server into receiving their spoofed packets and executing them, the attacker is successful in hijacking the session.

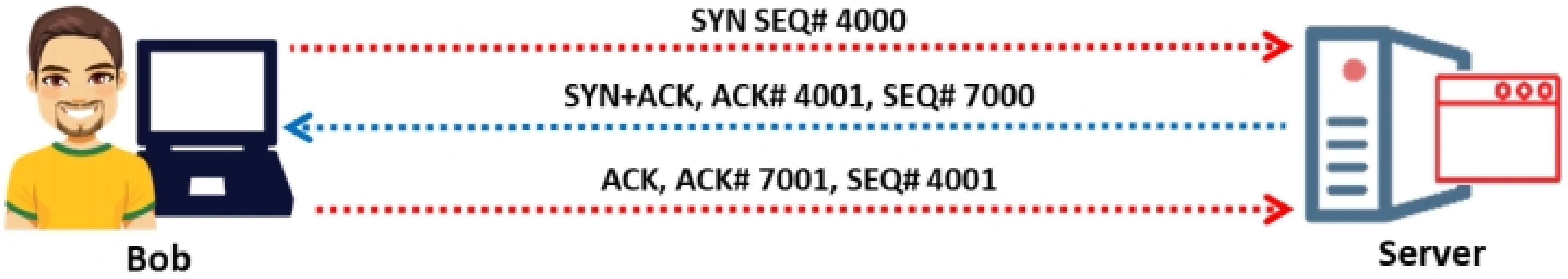
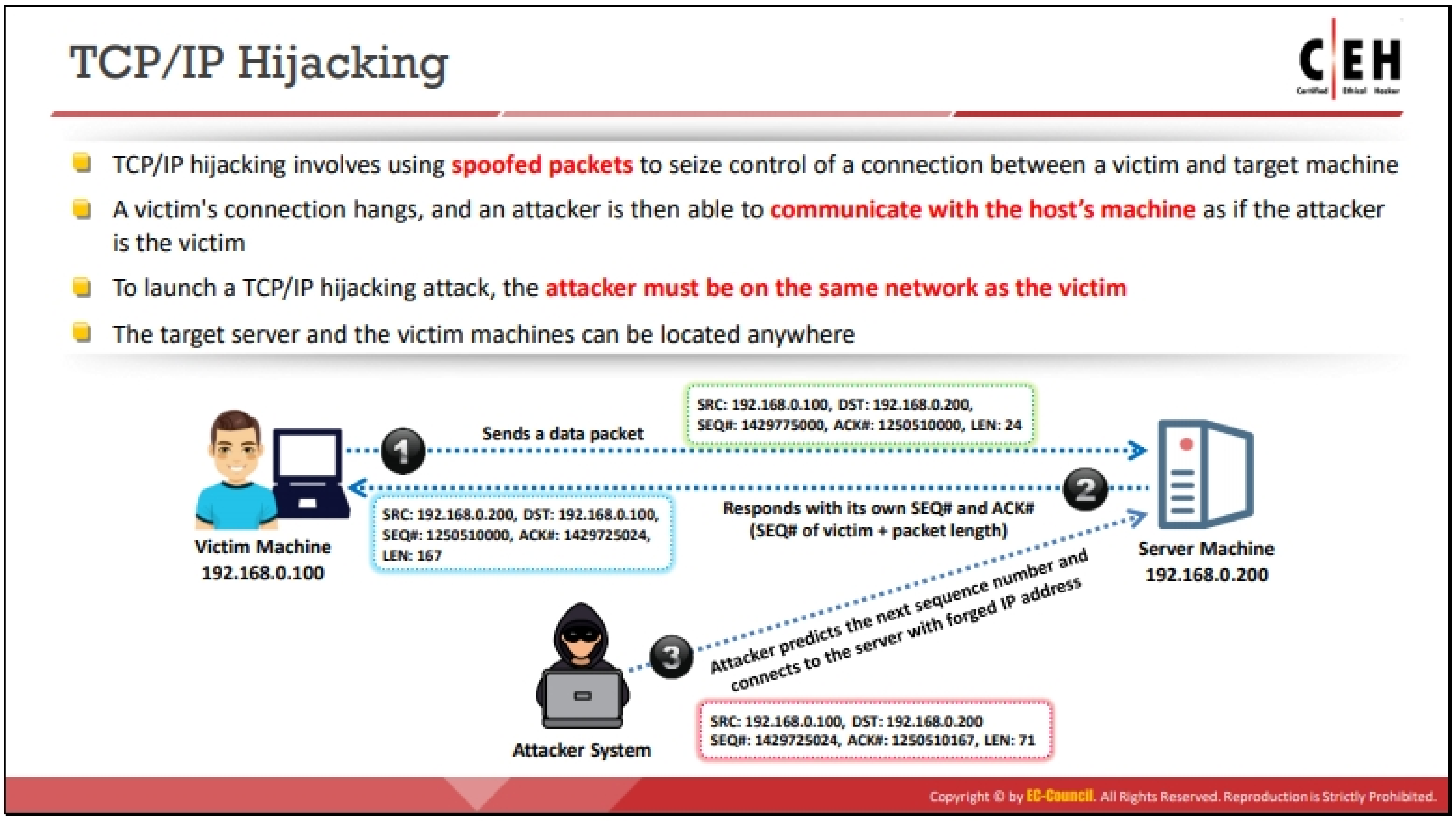


Figure 11.21: Three-way handshake

The three-way handshake shown in the above figure involves the following steps.

1. Bob initiates a connection with the server and sends a packet to the server with the SYN flag set.
2. The server receives this packet and sends a packet with the SYN + ACK flag and an initial sequence number (ISN) for the server.
3. Bob sets the ACK flag acknowledging the receipt of the packet and increments the sequence number by 1.
4. The two machines have successfully established a session.

If the attacker can anticipate the next sequence number and ACK number that Bob will send, they can spoof Bob's address and start communication with the server.



TCP/IP Hijacking

- TCP/IP hijacking involves using **spoofed packets** to seize control of a connection between a victim and target machine
- A victim's connection hangs, and an attacker is then able to **communicate with the host's machine** as if the attacker is the victim
- To launch a TCP/IP hijacking attack, the **attacker must be on the same network as the victim**
- The target server and the victim machines can be located anywhere

1
Sends a data packet
2
Responds with its own SEQ# and ACK#
(SEQ# of victim + packet length)
3
Attacker predicts the next sequence number and connects to the server with forged IP address
Victim Machine
192.168.0.100
Server Machine
192.168.0.200
Attacker System

SRC: 192.168.0.100, DST: 192.168.0.200,
SEQ#: 1429775000, ACK#: 1250510000, LEN: 24

SRC: 192.168.0.200, DST: 192.168.0.100,
SEQ#: 1250510000, ACK#: 1429725024,
LEN: 167

SRC: 192.168.0.100, DST: 192.168.0.200
SEQ#: 1429725024, ACK#: 1250510167, LEN: 71

TCP/IP Hijacking

In TCP/IP hijacking, an attacker intercepts an established connection between two communicating parties by using spoofed packets and then pretends to be one of those parties. In this approach, the attacker uses spoofed packets to redirect the TCP traffic to their own machine. Once this is successful, the victim's connection hangs, and the attacker is able to communicate with the host's machine on behalf of the victim. To launch a TCP/IP hijacking attack, both the victim and attacker must be on the same network. The target server and the victim machines can be located anywhere. By using this technique, an attacker can easily attack systems that use one-time passwords. As illustrated in the below figure, TCP/IP hijacking involves the following processes.

- The hacker sniffs the communication between the victim and host to obtain the victim's ISN.
- By using this ISN, the attacker sends a spoofed packet from the victim's IP address to the host system.
- The host machine responds to the victim, assuming that the packet arrived from it. This increments the sequence number.

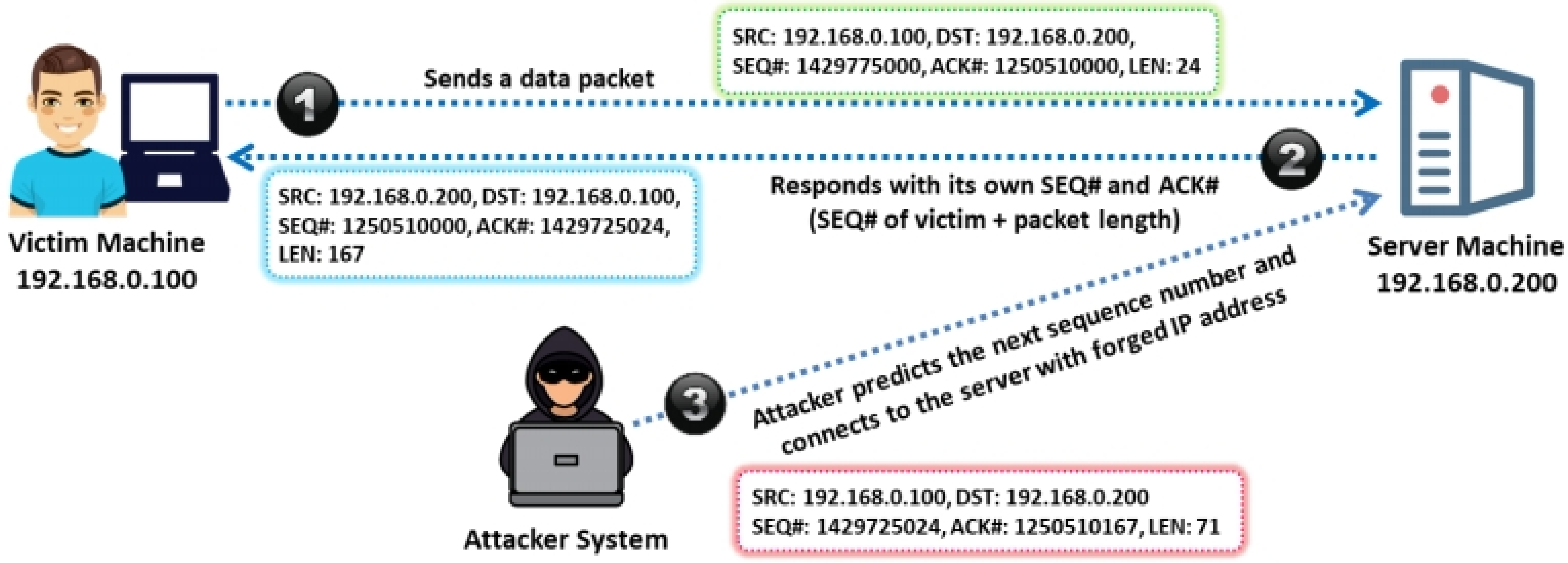
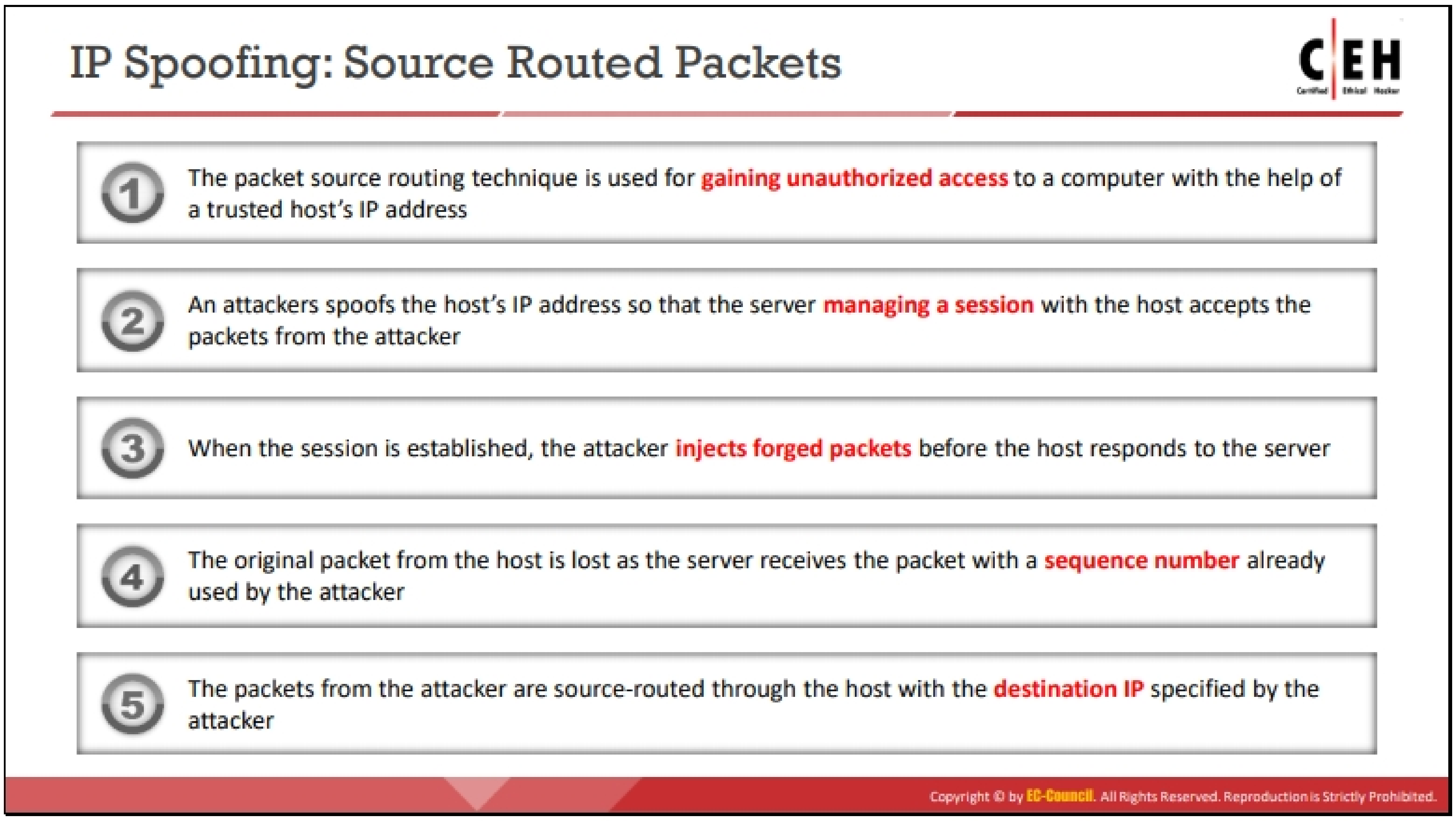

SRC: 192.168.0.100, DST: 192.168.0.200,
SEQ#: 1429775000, ACK#: 1250510000, LEN: 24
1
Sends a data packet
2
Responds with its own SEQ# and ACK#
(SEQ# of victim + packet length)
SRC: 192.168.0.200, DST: 192.168.0.100,
SEQ#: 1250510000, ACK#: 1429725024,
LEN: 167
Victim Machine
192.168.0.100
Server Machine
192.168.0.200
3
Attacker predicts the next sequence number and
connects to the server with forged IP address
Attacker System
SRC: 192.168.0.100, DST: 192.168.0.200
SEQ#: 1429725024, ACK#: 1250510167, LEN: 71

Figure 11.22: TCP/IP hijacking process

TCP/IP hijacking is performed through the following steps.

- The attacker sniffs the victim's connection and uses the victim's IP address to send a spoofed packet with the predicted sequence number.
- The receiver processes the spoofed packet, increments the sequence number, and sends an acknowledgement to the victim's IP address.
- The victim machine is unaware of the spoofed packet. Therefore, it ignores the receiver machine's ACK packet and turns off sequence number count.
- Consequently, the receiver receives packets with the incorrect sequence number.
- The attacker forces the victim's connection with the receiver machine into a desynchronized state.
- The attacker tracks sequence numbers and continuously spoofs packets that originate from the victim's IP address.
- The attacker continues to communicate with the receiver machine, while the victim's connection hangs.



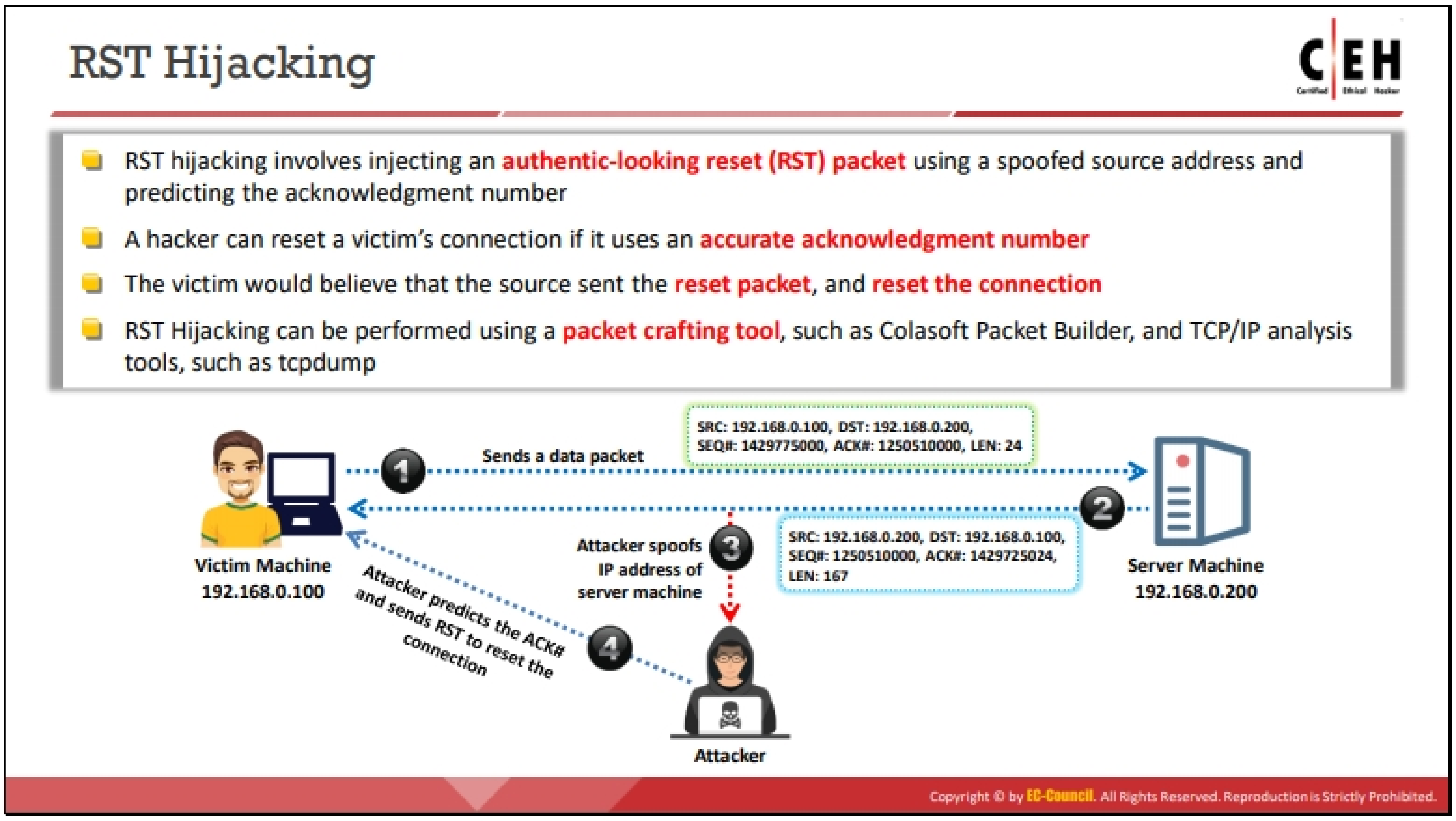
IP Spoofing: Source Routed Packets

1. The packet source routing technique is used for **gaining unauthorized access** to a computer with the help of a trusted host's IP address
2. An attackers spoofs the host's IP address so that the server **managing a session** with the host accepts the packets from the attacker
3. When the session is established, the attacker **injects forged packets** before the host responds to the server
4. The original packet from the host is lost as the server receives the packet with a **sequence number** already used by the attacker
5. The packets from the attacker are source-routed through the host with the **destination IP** specified by the attacker

IP Spoofing: Source Routed Packets

Source routed packets are useful in gaining unauthorized access to a computer with the help of a trusted host's IP address. This type of hijack allows attackers to create their own acceptable packets to insert into a TCP session. First, an attacker spoofs a trusted host's IP address so that the server managing a session with the host accepts the packets from the attacker. The packets are source routed; therefore, the sender specifies the path for packets from the source to the destination IP. By using this source-routing technique, attackers fool the server into believing that it is communicating with the user.

After spoofing the IP address successfully, the hijacker alters the sequence and acknowledgment numbers. Once these numbers are changed, the attacker injects forged packets into the TCP session before the client can respond. This leads to a desynchronized state because there the sequence and ACK numbers are not synchronized. The original packets are lost, and the server receives a packet with the new ISN. These packets are source routed to a patched destination IP address specified by the attacker.



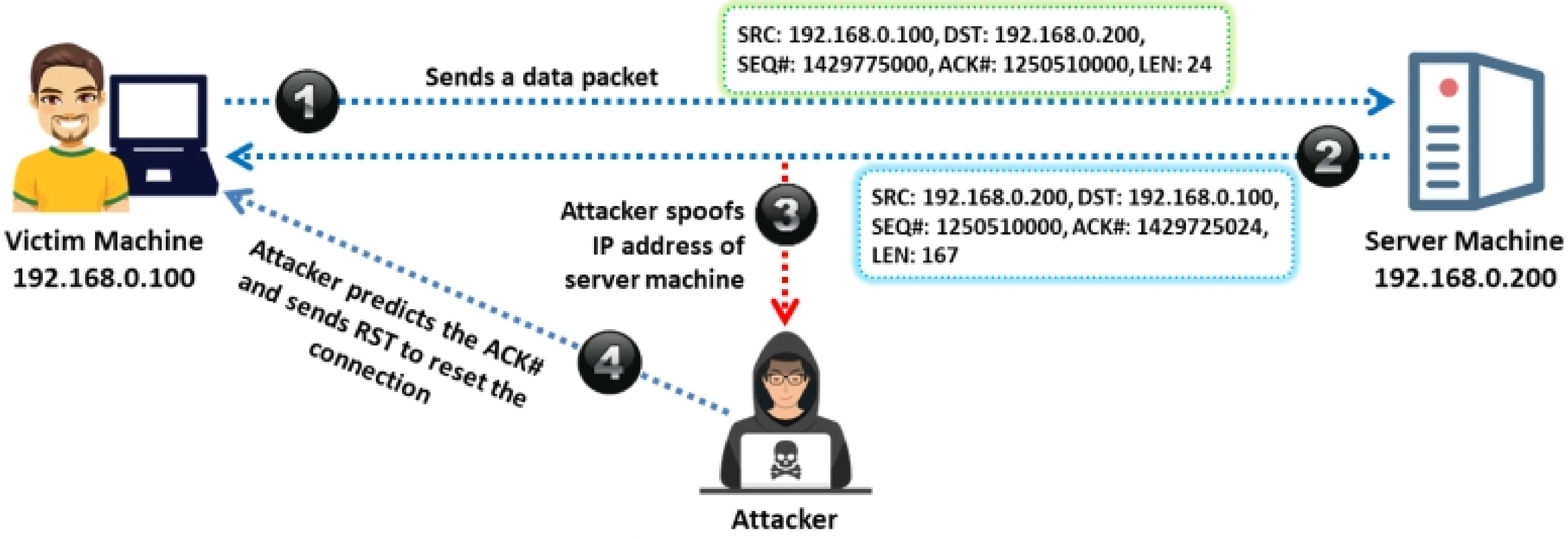
RST Hijacking

- RST hijacking involves injecting an **authentic-looking reset (RST) packet** using a spoofed source address and predicting the acknowledgment number
- A hacker can reset a victim's connection if it uses an **accurate acknowledgment number**
- The victim would believe that the source sent the **reset packet**, and **reset the connection**
- RST Hijacking can be performed using a **packet crafting tool**, such as Colasoft Packet Builder, and TCP/IP analysis tools, such as tcpdump

Sends a data packet
SRC: 192.168.0.100, DST: 192.168.0.200,
SEQ#: 1429775000, ACK#: 1250510000, LEN: 24
1
2
SRC: 192.168.0.200, DST: 192.168.0.100,
SEQ#: 1250510000, ACK#: 1429725024,
LEN: 167
3
Attacker spoofs IP address of server machine
4
Attacker predicts the ACK# and sends RST to reset the connection
Victim Machine
192.168.0.100
Server Machine
192.168.0.200
Attacker

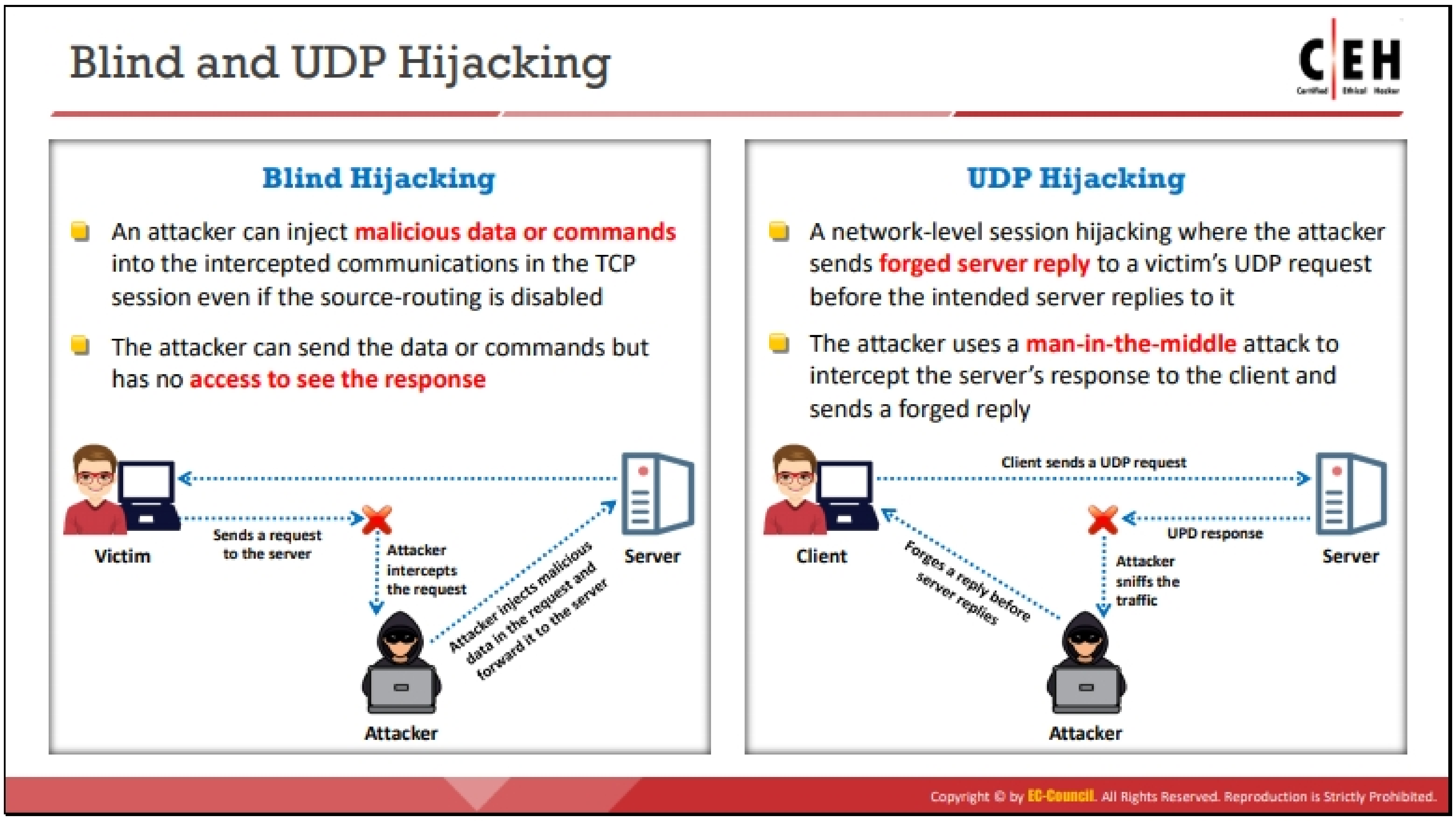
RST Hijacking

RST hijacking involves injecting an authentic-looking reset (RST) packet by using a spoofed source IP address and predicting the acknowledgment number. The hacker can reset the victim's connection if it uses an accurate acknowledgment number. The victim believes that the source has sent the reset packet and resets the connection. RST hijacking can be performed using packet-crafting tools such as Colasoft Packet Builder and TCP/IP analysis tools such as tcpdump.



Sends a data packet
SRC: 192.168.0.100, DST: 192.168.0.200,
SEQ#: 1429775000, ACK#: 1250510000, LEN: 24
1
2
SRC: 192.168.0.200, DST: 192.168.0.100,
SEQ#: 1250510000, ACK#: 1429725024,
LEN: 167
3
Attacker spoofs IP address of server machine
4
Attacker predicts the ACK# and sends RST to reset the connection
Victim Machine
192.168.0.100
Server Machine
192.168.0.200
Attacker

Figure 11.23: RST hijacking process



Blind and UDP Hijacking

Blind Hijacking

- An attacker can inject **malicious data or commands** into the intercepted communications in the TCP session even if the source-routing is disabled
- The attacker can send the data or commands but has no **access to see the response**

Victim
Sends a request to the server
Attacker intercepts the request
Attacker injects malicious data in the request and forward it to the server
Server
Attacker

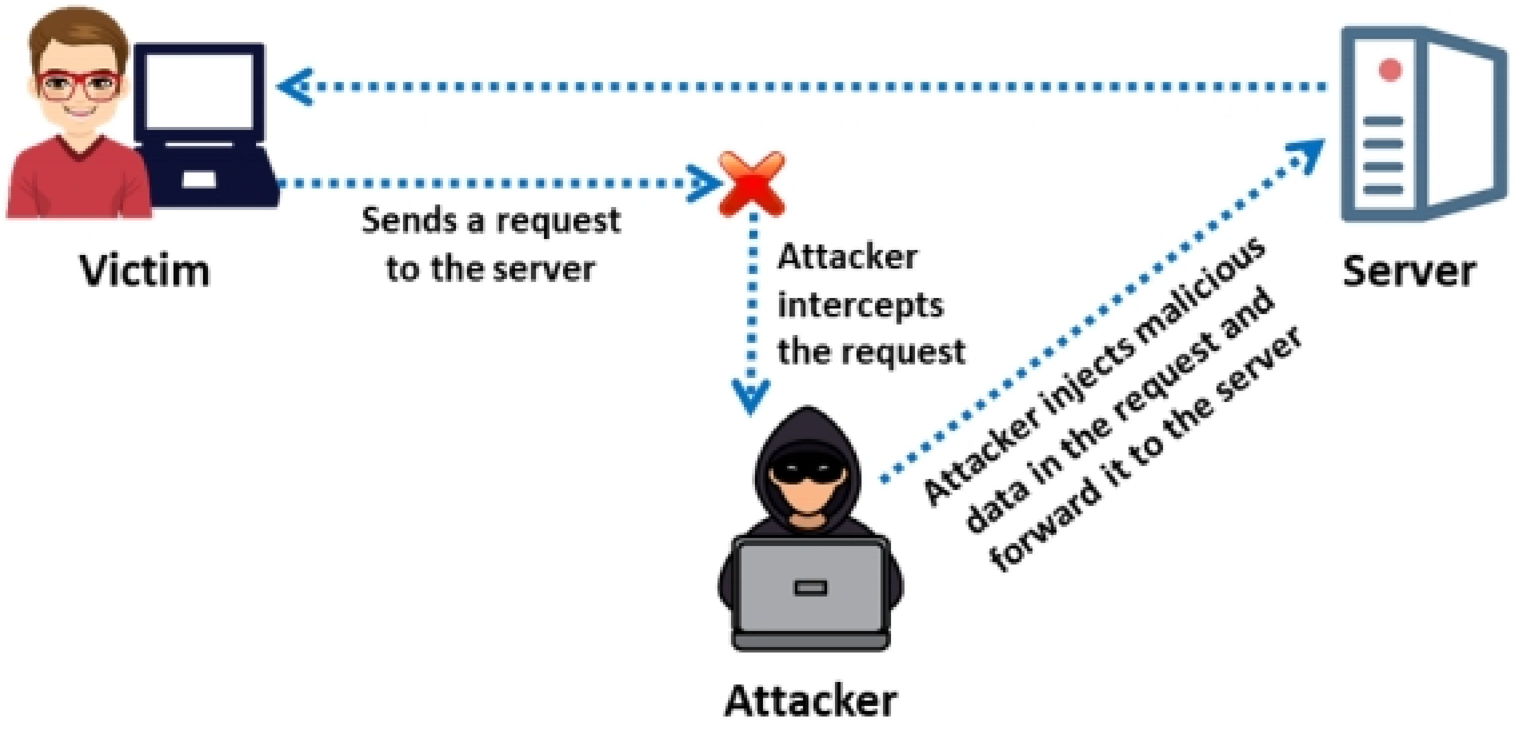
UDP Hijacking

- A network-level session hijacking where the attacker sends **forged server reply** to a victim's UDP request before the intended server replies to it
- The attacker uses a **man-in-the-middle** attack to intercept the server's response to the client and sends a forged reply

Client sends a UDP request
Client
UPD response
Server
Forges a reply before server replies
Attacker sniffs the traffic
Attacker

Blind Hijacking

In blind hijacking, an attacker can inject malicious data or commands into intercepted communications in a TCP session, even if the victim disables source routing. For this purpose, the attacker must correctly guess the next ISN of a computer attempting to establish a connection. Although the attacker can send malicious data or a command, such as a password setting to allow access from another location on the network, the attacker cannot view the response. To be able to view the response, an MITM attack is a much better option.



Victim
Sends a request to the server
Attacker intercepts the request
Attacker injects malicious data in the request and forward it to the server
Server
Attacker

Figure 11.24: Blind hijacking process

UDP Hijacking

The User Datagram Protocol (UDP) does not use packet sequencing or synchronizing. Therefore, a UDP session can be attacked more easily than a TCP session. Because UDP is connectionless, it is easy to modify data without the victim noticing. In network level session hijack, the hijacker forges a server reply to a client UDP request before the server can respond. Thus, the attacker takes control of the session. No packets are exchanged between the server and client, because the server's sequence number fails to match the client's acknowledgement number.

The server's reply can be easily restricted if sniffing is used. An MITM attack in UDP hijacking can minimize the task of the attacker because it can stop the server's reply from reaching the client in the first place.

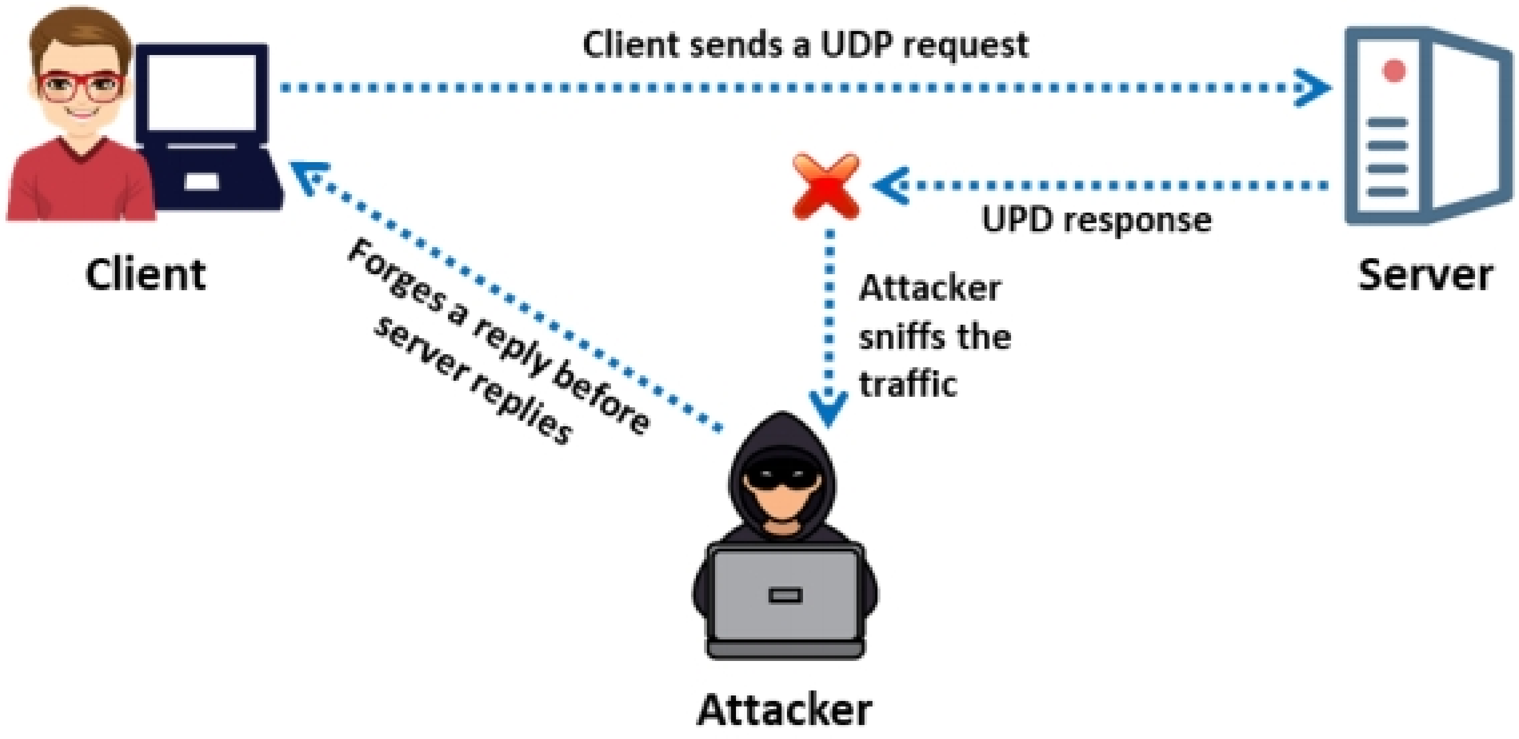
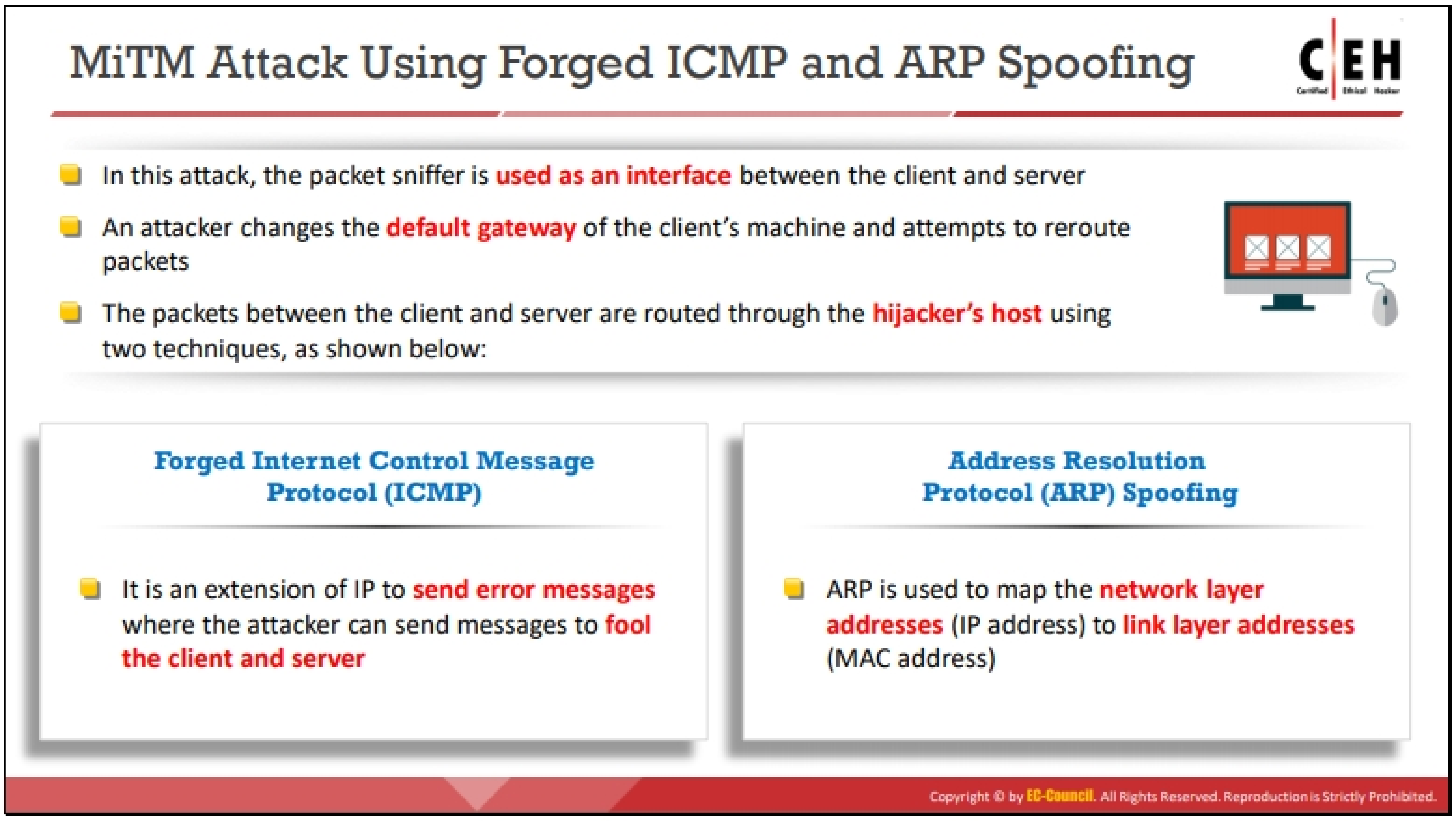

Client sends a UDP request
Client
Server
UPD response
Attacker sniffs the traffic
Forges a reply before server replies
Attacker

Figure 11.25: Hijacking a UDP session



MiTM Attack Using Forged ICMP and ARP Spoofing

- In this attack, the packet sniffer is **used as an interface** between the client and server
- An attacker changes the **default gateway** of the client's machine and attempts to reroute packets
- The packets between the client and server are routed through the **hijacker's host** using two techniques, as shown below:

Forged Internet Control Message Protocol (ICMP)

- It is an extension of IP to **send error messages** where the attacker can send messages to **fool the client and server**

Address Resolution Protocol (ARP) Spoofing

- ARP is used to map the **network layer addresses** (IP address) to **link layer addresses** (MAC address)

MITM Attack Using Forged ICMP and ARP Spoofing

An MITM attack uses a packet sniffer to intercept communication between a client and server. The attacker changes the default gateway of the client's machine and attempts to reroute packets. The packets between the client and server are routed through the hijacker's host by using the following two techniques.

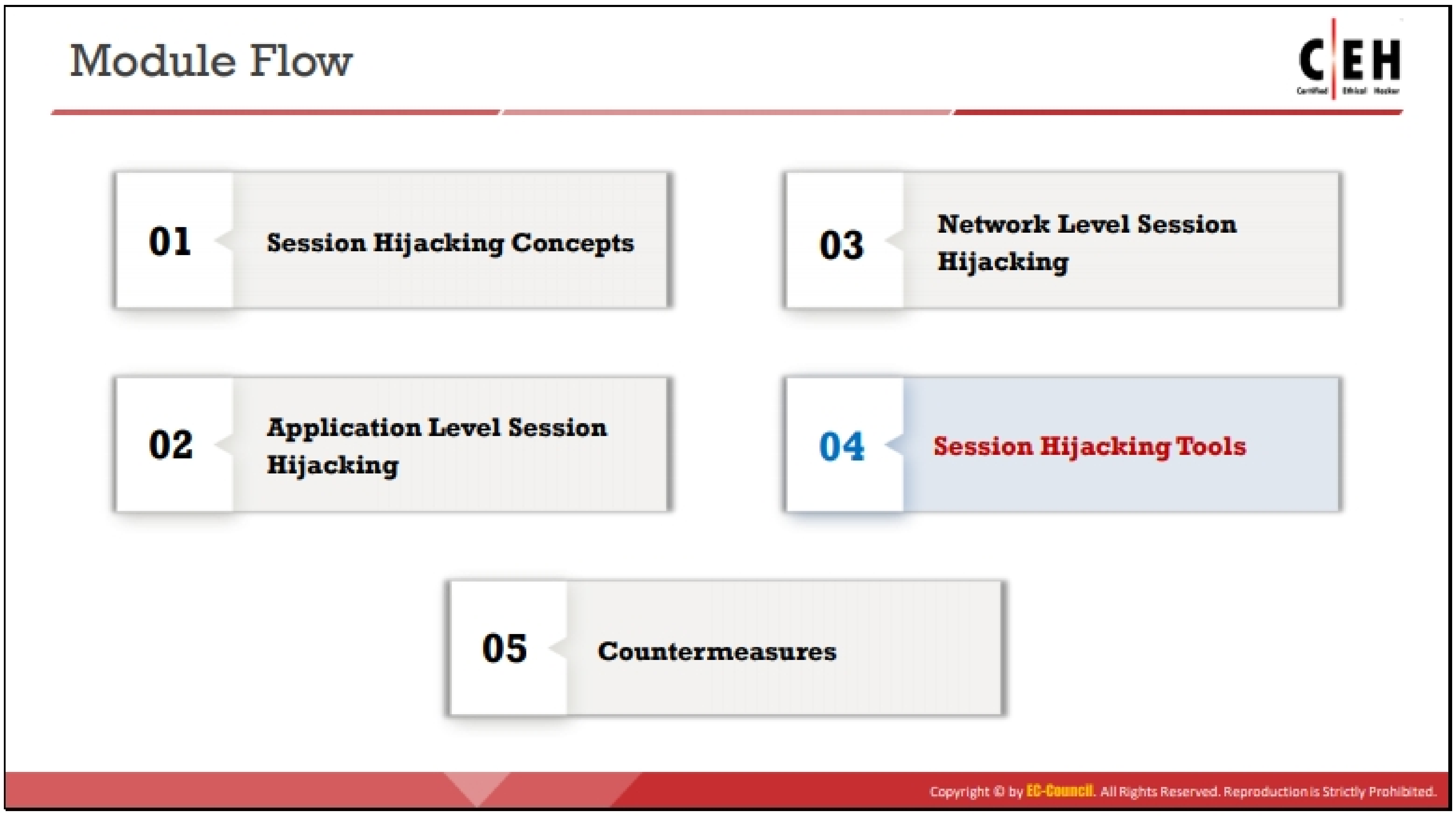
- **Forged Internet Control Message Protocol (ICMP)**

  The Internet Control Message Protocol (ICMP) is an extension of IP used to send error messages. An attacker can use ICMP to send messages to fool the client and server. In this technique, ICMP packets are forged to redirect traffic between the client and host through the hijacker's host. The hacker's packets send error messages indicating problems in processing packets through the original connection. This fools the server and client into routing through the hijacker's path instead.

- **Address Resolution Protocol (ARP) Spoofing**

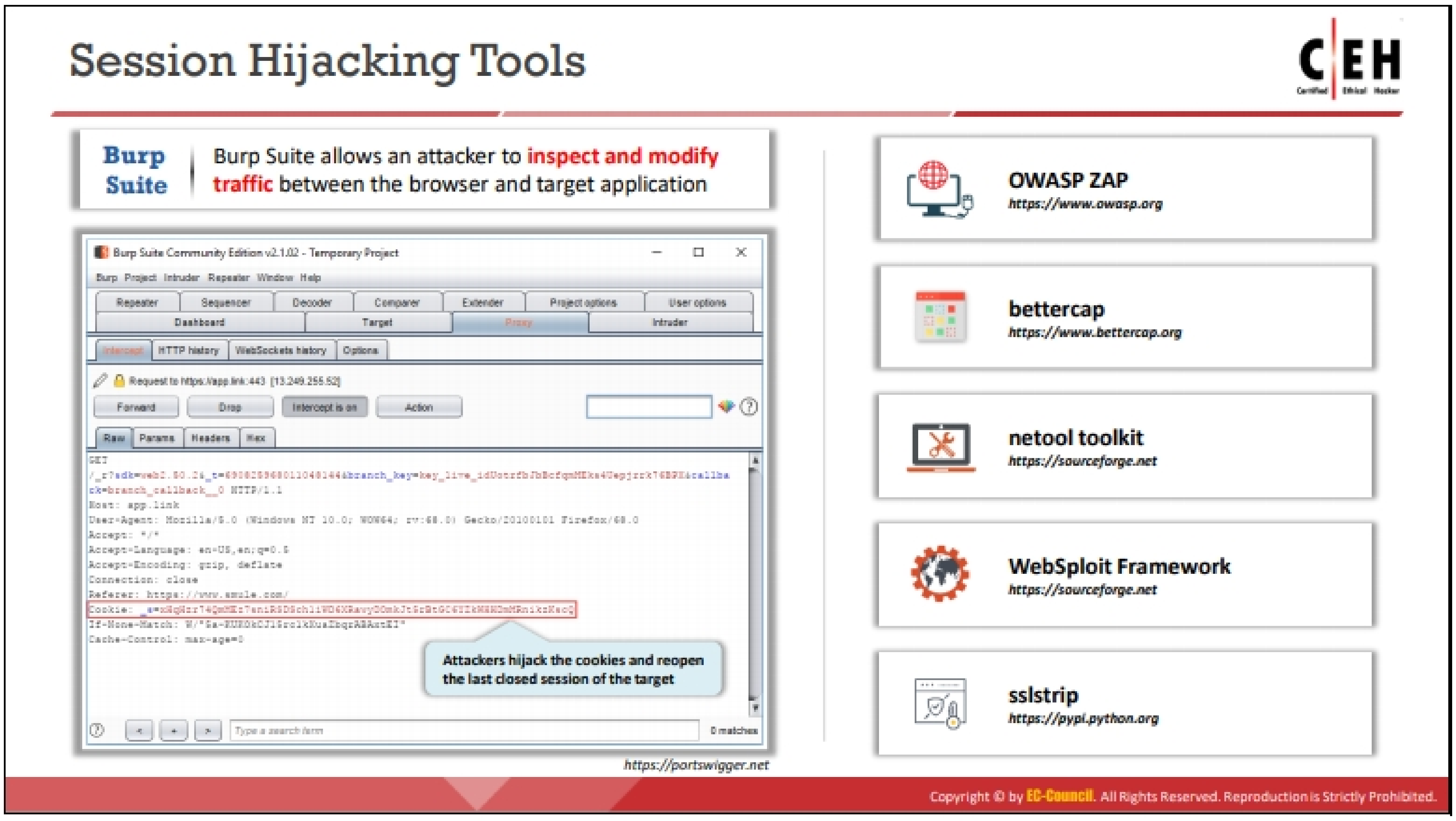
  Hosts use Address Resolution Protocol (ARP) tables to map local network layer addresses (IP addresses) to hardware addresses or MAC addresses. This technique involves fooling the host by broadcasting the ARP request and changing its ARP tables by sending forged ARP replies. The attacker sends forged ARP replies that update the ARP tables of the host that is broadcasting ARP requests. This routes the traffic to the attacker's host instead of the legitimate IP address.

In both techniques, an attacker routes the packets in transit between the client and server through their machine.



Module Flow

01	Session Hijacking Concepts
02	Application Level Session Hijacking
03	Network Level Session Hijacking
04	**Session Hijacking Tools**
05	Countermeasures



Session Hijacking Tools

Burp Suite | Burp Suite allows an attacker to **inspect and modify traffic** between the browser and target application

Burp Suite Community Edition v2.1.02 - Temporary Project
Burp Project Intruder Repeater Window Help
Repeater Sequencer Decoder Comparer Extender Project options User options
Dashboard Target Proxy Intruder
Intercept HTTP history WebSockets history Options
Request to https://app.link:443 [13.249.255.52]
Forward Drop Intercept is on Action
Raw Params Headers Hex
Type a search term
0 matches

Attackers hijack the cookies and reopen the last closed session of the target

https://portswigger.net

- **OWASP ZAP** *https://www.owasp.org*
- **bettercap** *https://www.bettercap.org*
- **netool toolkit** *https://sourceforge.net*
- **WebSploit Framework** *https://sourceforge.net*
- **sslstrip** *https://pypi.python.org*

Session Hijacking Tools

Attackers can use tools such as Burp Suite, OWASP ZAP, and bettercap to hijack a session between a client and server. This section discusses various tools that help perform session hijacking.

- **Burp Suite**

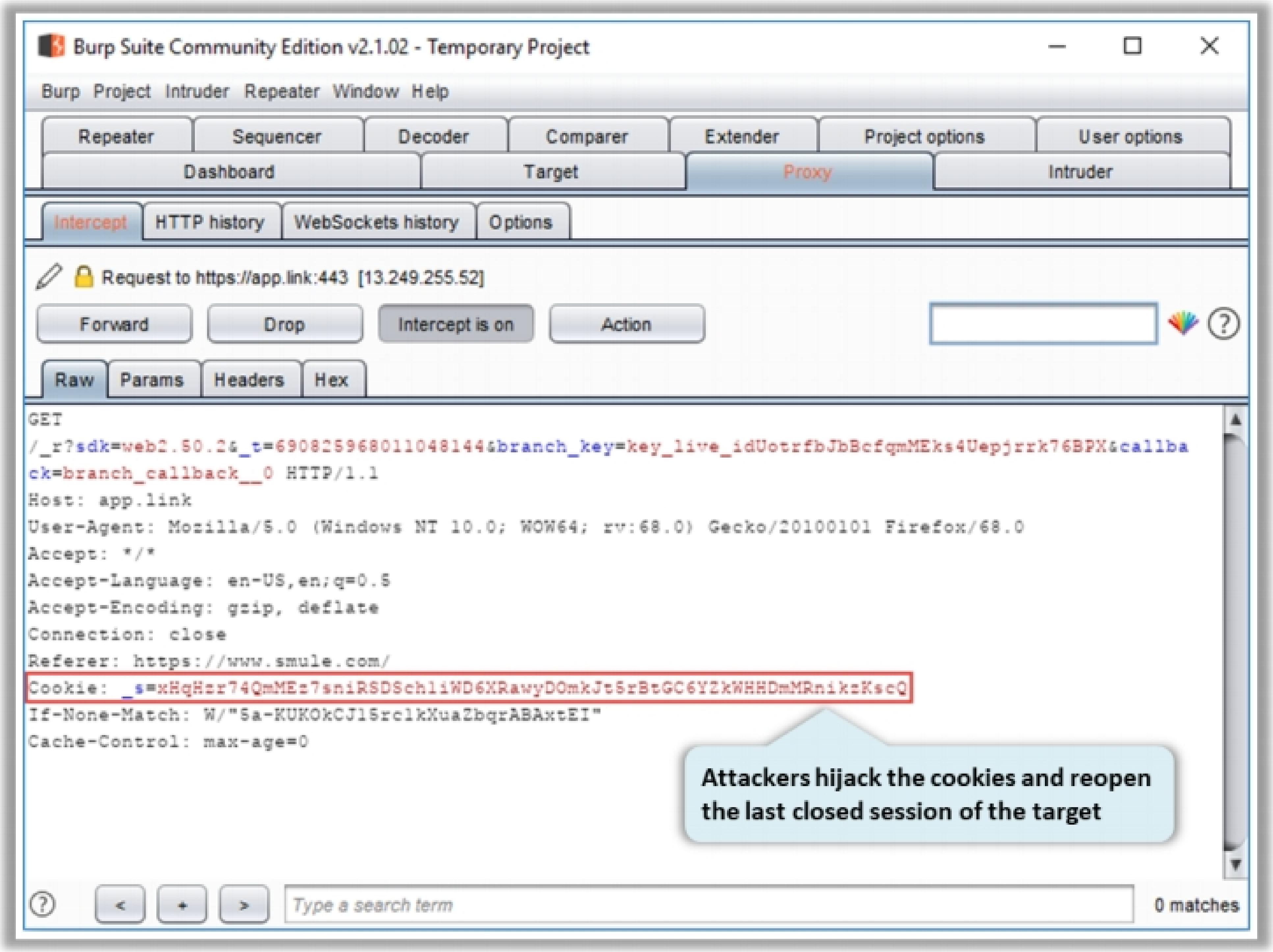
  Source: *https://portswigger.net*

Burp Suite is an integrated platform for the security testing of web applications. It allows attackers to inspect and modify traffic between a browser and target application.

Burp Suite contains the following key components.

- An intercepting proxy, which allows the user to inspect and modify traffic between their browser and the target application
- An application-aware spider that crawls content and functionality
- An advanced web application scanner that automates the detection of numerous types of vulnerability
- An intruder tool for performing powerful customized attacks to find and exploit unusual vulnerabilities
- A repeater tool for manipulating and resending individual requests
- A sequencer tool for testing the randomness of session tokens
- The CSRF PoC Generator function, which generates proof-of-concept cross-site request forgery (CSRF) attacks for a given request

As shown in the figure, attackers can use this tool to capture a session cookie of a victim and later use that cookie to reopen a closed session of the victim.


Burp Suite Community Edition v2.1.02 - Temporary Project
Burp Project Intruder Repeater Window Help
Repeater Sequencer Decoder Comparer Extender Project options User options
Dashboard Target Proxy Intruder
Intercept HTTP history WebSockets history Options
Request to https://app.link:443 [13.249.255.52]
Forward Drop Intercept is on Action
Raw Params Headers Hex
Type a search term
0 matches

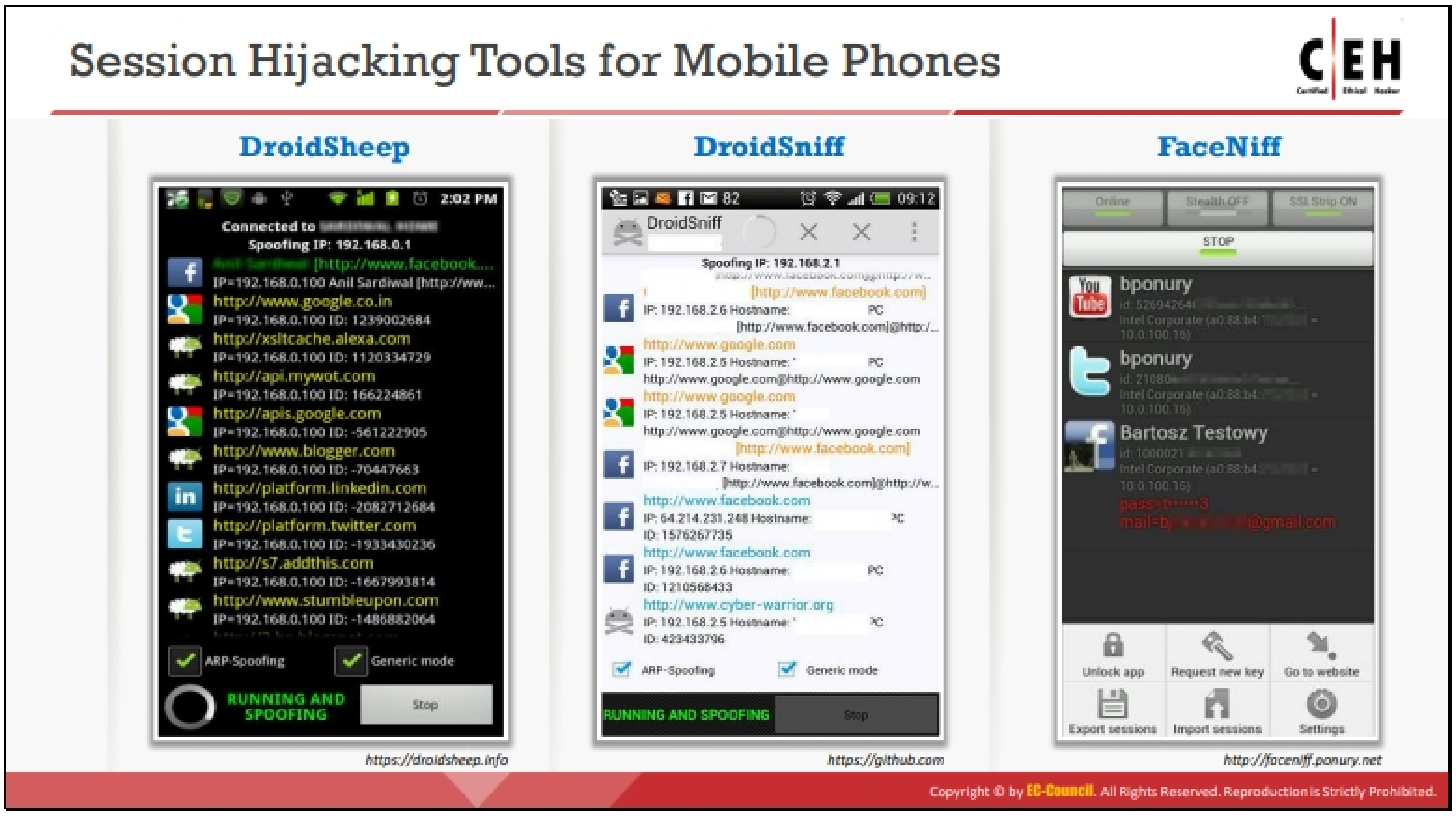
```
GET
/_r?sdk=web2.50.2&_t=690825968011048144&branch_key=key_live_idUotrfbJbBcfqmMEks4Uepjrrk76BPX&callba
ck=branch_callback__0 HTTP/1.1
Host: app.link
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: https://www.smule.com/
Cookie: _s=xHqHzr74QmMEz7sniRSDSchliWD6XRawyDOmkJt5rBtGC6YZkWHHDmMRnikzKscQ
If-None-Match: W/"5a-KUKOkCJl5rclkXuaZbqrABAxtEI"
Cache-Control: max-age=0
```

Attackers hijack the cookies and reopen the last closed session of the target

Figure 11.26: Screenshot of Burp Suite

The following are some additional session hijacking tools:

- OWASP ZAP (*https://www.owasp.org*)
- bettercap (*https://www.bettercap.org*)
- netool toolkit (*https://sourceforge.net*)
- WebSploit Framework (*https://sourceforge.net*)
- sslstrip (*https://pypi.python.org*)



Session Hijacking Tools for Mobile Phones

DroidSheep

```
2:02 PM
Connected to [redacted]
Spoofing IP: 192.168.0.1
[redacted] [http://www.facebook....
IP=192.168.0.100 Anil Sardiwal [http://ww...
http://www.google.co.in
IP=192.168.0.100 ID: 1239002684
http://xsltcache.alexa.com
IP=192.168.0.100 ID: 1120334729
http://api.mywot.com
IP=192.168.0.100 ID: 166224861
http://apis.google.com
IP=192.168.0.100 ID: -561222905
http://www.blogger.com
IP=192.168.0.100 ID: -70447663
http://platform.linkedin.com
IP=192.168.0.100 ID: -2082712684
http://platform.twitter.com
IP=192.168.0.100 ID: -1933430236
http://s7.addthis.com
IP=192.168.0.100 ID: -1667993814
http://www.stumbleupon.com
IP=192.168.0.100 ID: -1486882064
```

https://droidsheep.info

DroidSniff

```
82                    09:12
DroidSniff
Spoofing IP: 192.168.2.1
[redacted] [http://www.facebook.com]
IP: 192.168.2.6 Hostname: [redacted] PC
[redacted] [http://www.facebook.com]@http:/...
http://www.google.com
IP: 192.168.2.5 Hostname: [redacted] PC
http://www.google.com@http://www.google.com
http://www.google.com
IP: 192.168.2.5 Hostname: [redacted]
http://www.google.com@http://www.google.com
[redacted] [http://www.facebook.com]
IP: 192.168.2.7 Hostname: [redacted]
[redacted] [http://www.facebook.com]@http://w...
http://www.facebook.com
IP: 64.214.231.248 Hostname: [redacted] PC
ID: 1576267735
http://www.facebook.com
IP: 192.168.2.6 Hostname: [redacted] PC
ID: 1210568433
http://www.cyber-warrior.org
IP: 192.168.2.5 Hostname: [redacted] PC
ID: 423433796
```

https://github.com

FaceNiff

```
bponury
id: 52694264[redacted]
Intel Corporate (a0:88:b4:[redacted] -
10.0.100.16)
bponury
id: 21080[redacted]
Intel Corporate (a0:88:b4:[redacted] -
10.0.100.16)
Bartosz Testowy
id: 1000021[redacted]
Intel Corporate (a0:88:b4:[redacted] -
10.0.100.16)
pass=[redacted]3
mail=b[redacted]@gmail.com
```

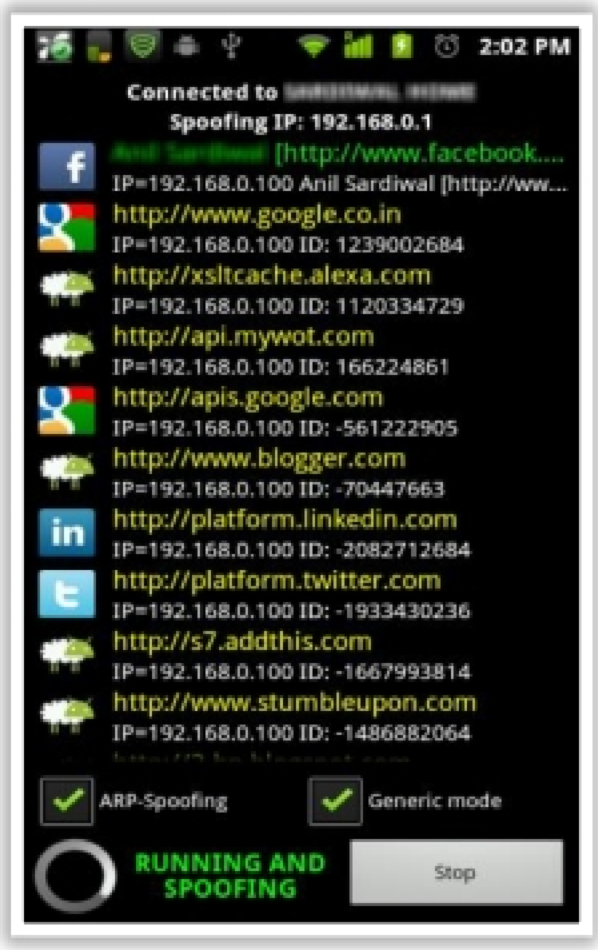
http://faceniff.ponury.net

Session Hijacking Tools for Mobile Phones

- **DroidSheep**

  Source: *https://droidsheep.info*

  The DroidSheep tool is used for session hijacking on Android devices connected to a common wireless network. It obtains the session ID of active users on the Wi-Fi network and uses it to access a website as an authorized user. A DroidSheep user can easily observe the activities of authorized users on websites. It can also hijack social accounts by obtaining the session ID.



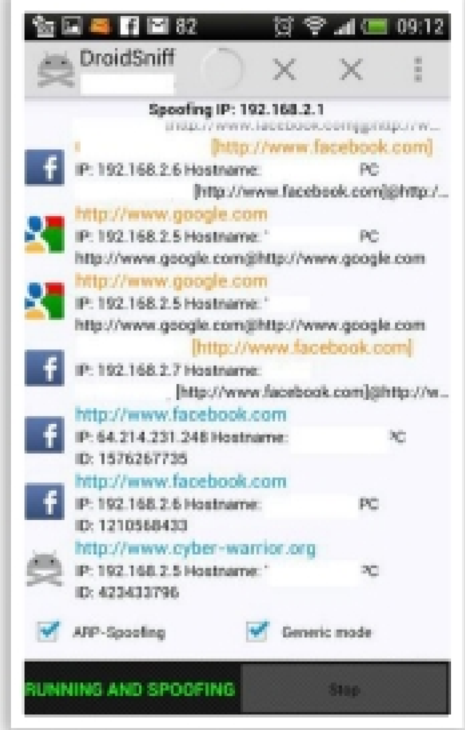
```
2:02 PM
Connected to [redacted]
Spoofing IP: 192.168.0.1
[redacted] [http://www.facebook....
IP=192.168.0.100 Anil Sardiwal [http://ww...
http://www.google.co.in
IP=192.168.0.100 ID: 1239002684
http://xsltcache.alexa.com
IP=192.168.0.100 ID: 1120334729
http://api.mywot.com
IP=192.168.0.100 ID: 166224861
http://apis.google.com
IP=192.168.0.100 ID: -561222905
http://www.blogger.com
IP=192.168.0.100 ID: -70447663
http://platform.linkedin.com
IP=192.168.0.100 ID: -2082712684
http://platform.twitter.com
IP=192.168.0.100 ID: -1933430236
http://s7.addthis.com
IP=192.168.0.100 ID: -1667993814
http://www.stumbleupon.com
IP=192.168.0.100 ID: -1486882064
```

Figure 11.27: Screenshot of DroidSheep

- **DroidSniff**

  Source: *https://github.com*

  DroidSniff is an Android app for security analysis in wireless networks that can capture Facebook, Twitter, LinkedIn, and other accounts. This tool is used for testing the security of user accounts. It identifies the poor security properties of network connections without encryption.



```
Spoofing IP: 192.168.2.1
[redacted] http://www.facebook.com@http://w...
[redacted] [http://www.facebook.com]
IP: 192.168.2.6 Hostname: [redacted] PC
[redacted] [http://www.facebook.com]@http:/...
http://www.google.com
IP: 192.168.2.5 Hostname: [redacted] PC
http://www.google.com@http://www.google.com
http://www.google.com
IP: 192.168.2.5 Hostname: [redacted]
http://www.google.com@http://www.google.com
[redacted] [http://www.facebook.com]
IP: 192.168.2.7 Hostname: [redacted]
[redacted] [http://www.facebook.com]@http://w...
http://www.facebook.com
IP: 64.214.231.248 Hostname: [redacted] PC
ID: 1576267735
http://www.facebook.com
IP: 192.168.2.6 Hostname: [redacted] PC
ID: 1210568433
http://www.cyber-warrior.org
IP: 192.168.2.5 Hostname: [redacted] PC
ID: 423433796
```

☑ ARP-Spoofing ☑ Generic mode

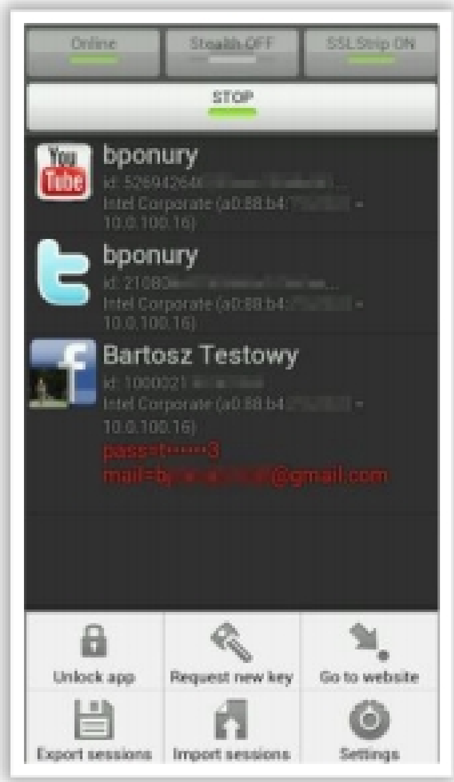
RUNNING AND SPOOFING

Figure 11.28: Screenshot of DroidSniff

- **FaceNiff**

  Source: *http://faceniff.ponury.net*

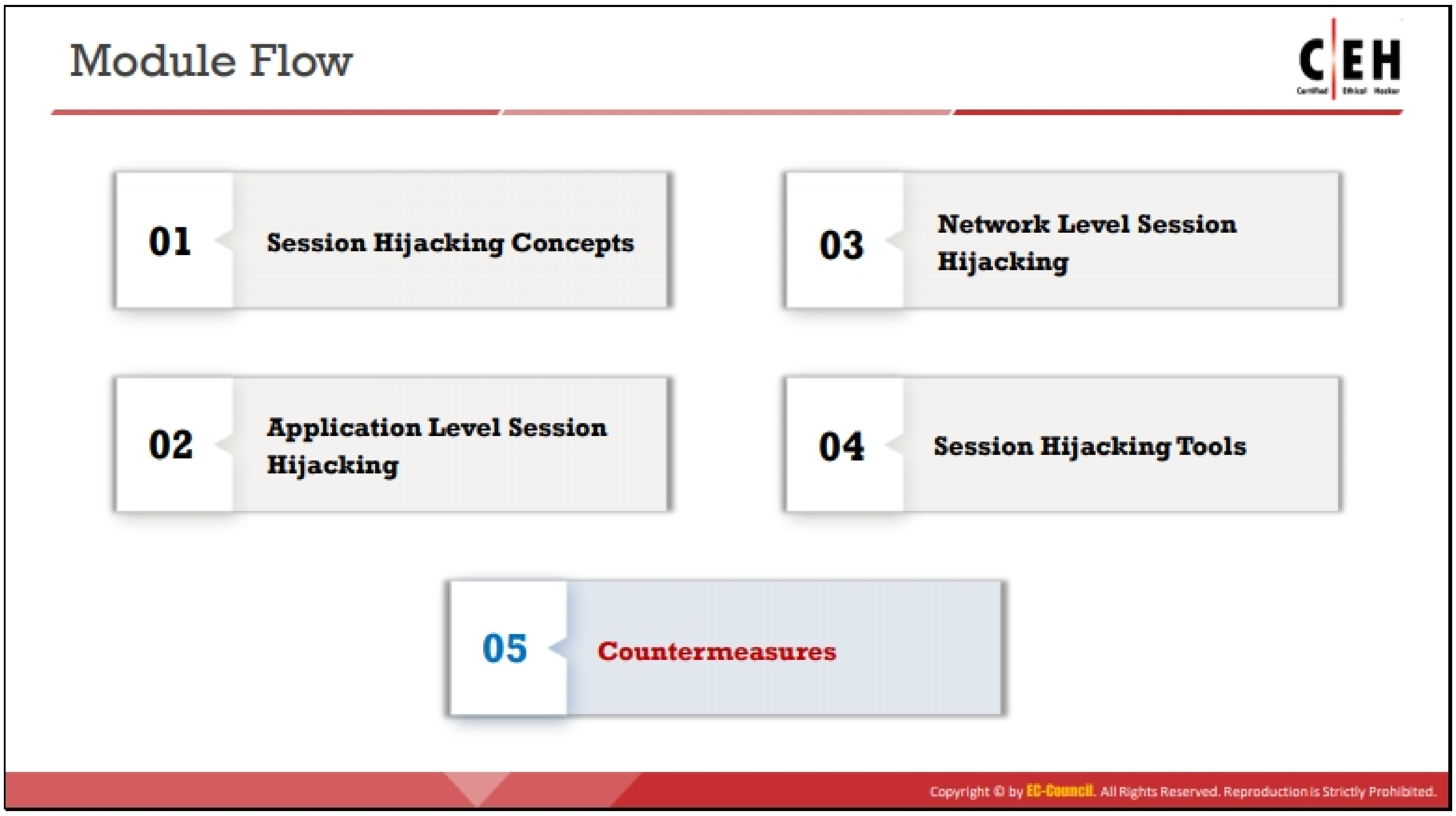
  FaceNiff is an Android app that allows a user to sniff and intercept web-session profiles over the WiFi network that the user's mobile device is connected to. Although FaceNiff can hijack sessions only when the WiFi network does not use the Extensible Authentication Protocol (EAP), it works on any private network, including open, Wired Equivalent Privacy (WEP), Wi-Fi Protected Access–pre-shared key (WPA-PSK), and WPA2-PSK networks.



Online | Stealth OFF | SSLStrip ON

```
bponury
id: 526942646[redacted]...
Intel Corporate (a0:88:b4:[redacted] -
10.0.100.16)
bponury
id: 21083[redacted]...
Intel Corporate (a0:88:b4:[redacted] -
10.0.100.16)
Bartosz Testowy
id: 1000021[redacted]
Intel Corporate (a0:88:b4:[redacted] -
10.0.100.16)
pass=t•••••3
mail=b[redacted]@gmail.com
```

Figure 11.29: Screenshot of FaceNiff



Module Flow

01 **Session Hijacking Concepts**

02 **Application Level Session Hijacking**

03 **Network Level Session Hijacking**

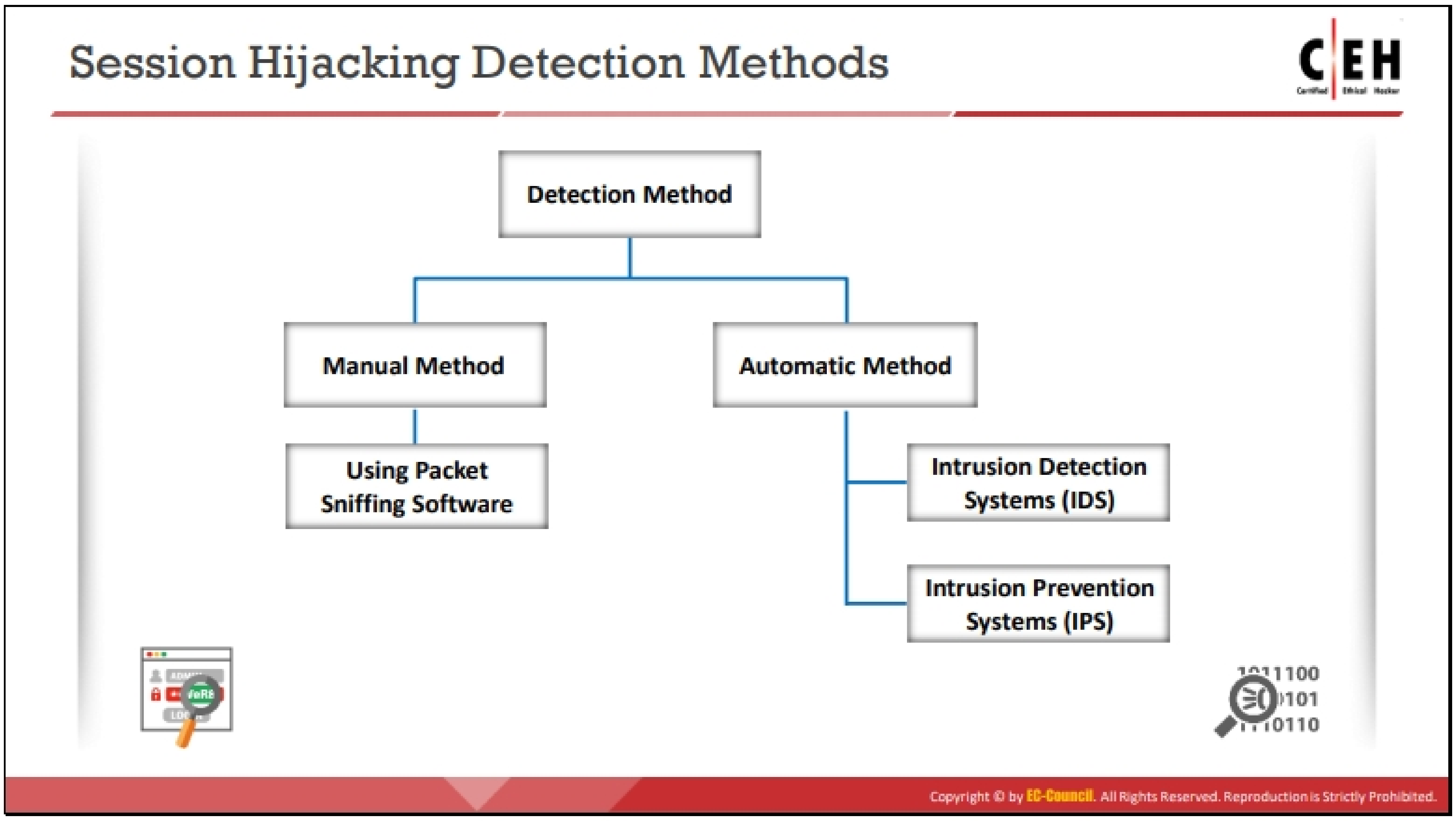
04 **Session Hijacking Tools**

05 **Countermeasures**

Countermeasures

In general, hijacking is a dangerous attack because the victim is at risk of identity theft, fraud, and loss of sensitive information. All networks using TCP/IP are vulnerable to the different types of session hijacking attacks discussed earlier. However, following best practices might protect against session hijacking attacks.

This section discusses session hijacking detection methods, session hijacking detection tools, various countermeasures to combat session hijacking attacks, and approaches causing vulnerability to session hijacking and their preventative solutions such as IP Security (IPsec).



Session Hijacking Detection Methods

Detection Method

Manual Method

Using Packet Sniffing Software

Automatic Method

Intrusion Detection Systems (IDS)

Intrusion Prevention Systems (IPS)

Session Hijacking Detection Methods

Session hijacking attacks are exceptionally difficult to detect, and users often overlook them unless the attacker causes severe damage.

The following are some symptoms of a session hijacking attack:

- A burst of network activity for some time, which decreases the system performance
- Busy servers resulting from requests sent by both the client and hijacker

Methods to detect session hijacking

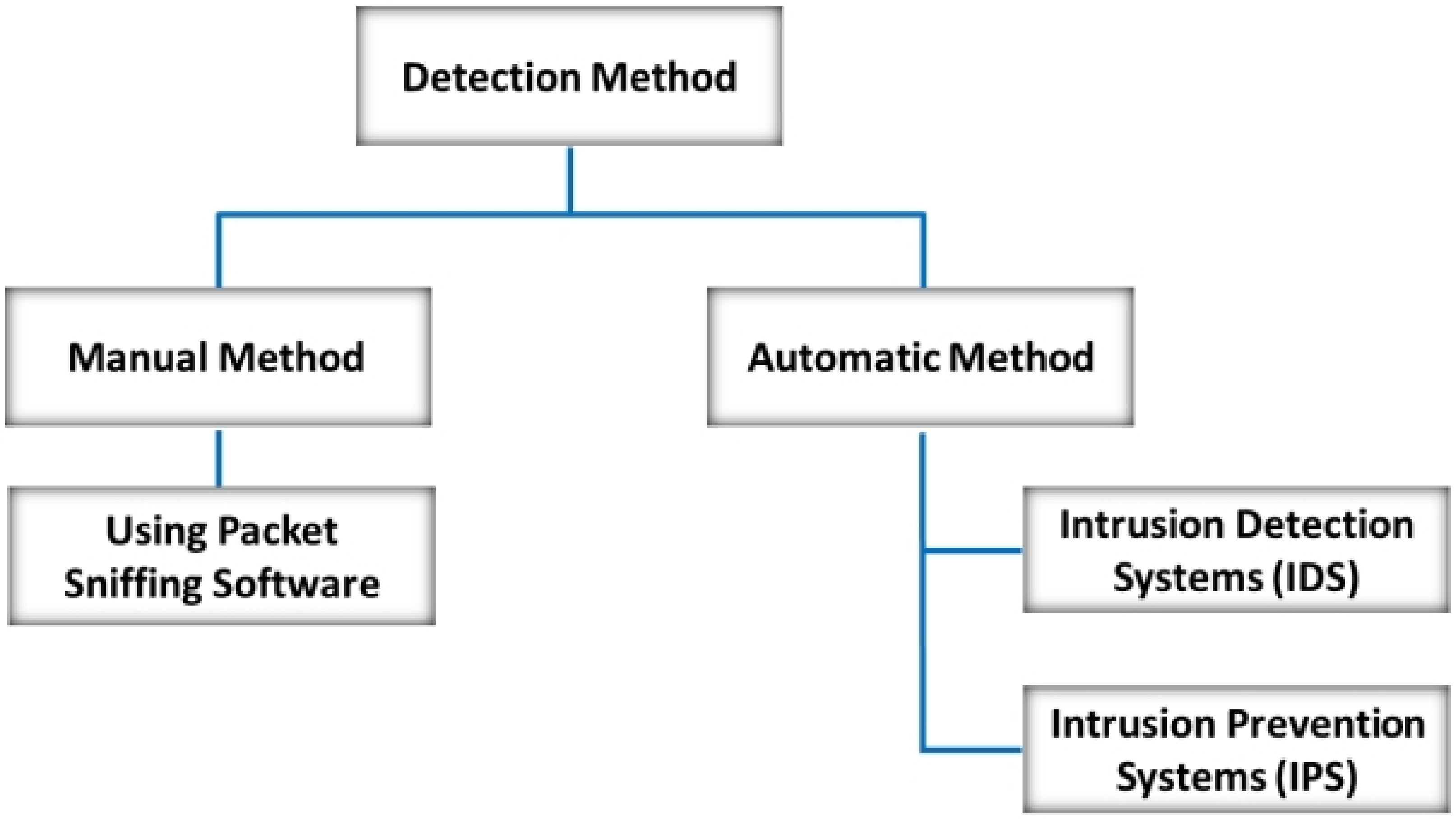

Detection Method
Manual Method
Automatic Method
Using Packet Sniffing Software
Intrusion Detection Systems (IDS)
Intrusion Prevention Systems (IPS)

Figure 11.30: Session hijacking detection methods

- **Manual Method**

  The manual method involves the use of packet sniffing software such as Wireshark and SteelCentral Packet Analyzer to monitor session hijacking attacks. The packet sniffer captures packets in transit across the network, which is then analyzed using various filtering tools.

  Forced ARP Entry

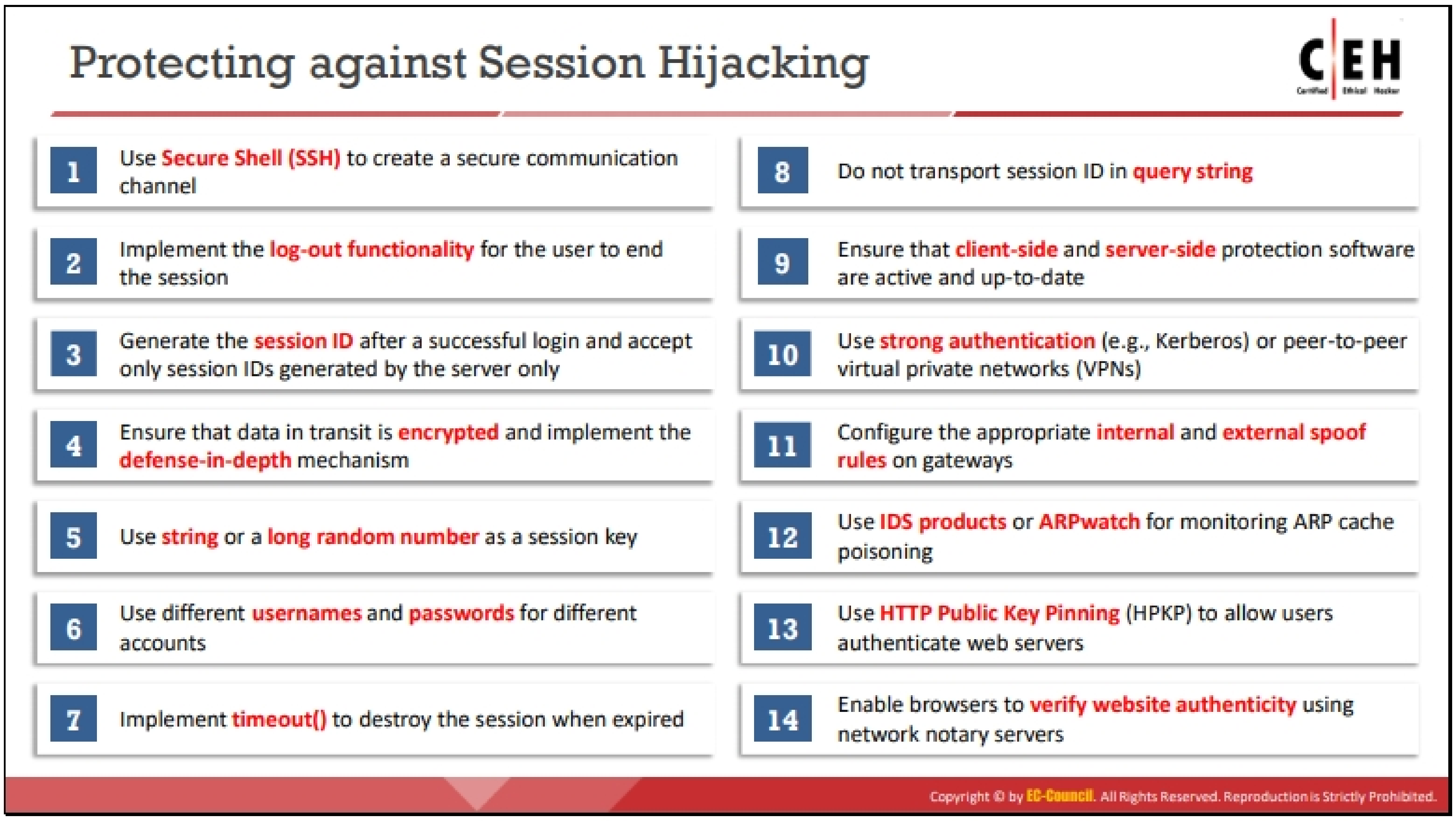
  A forced ARP entry involves replacing the MAC address of a compromised machine in the ARP cache of the server with a different one in order to restrict network traffic to the compromised machine.

  A forced ARP entry should be performed in the case of the following:

  - Repeated ARP updates
  - Frames sent between the client and server with different MAC addresses
  - ACK storms

- **Automatic Method**

  The automatic method involves the use of an intrusion detection systems (IDS) and intrusion prevention systems (IPS) to monitor incoming network traffic. If the packet matches any of the attack signatures in the internal database, the IDS generates an alert, whereas the IPS blocks the traffic from entering the database.



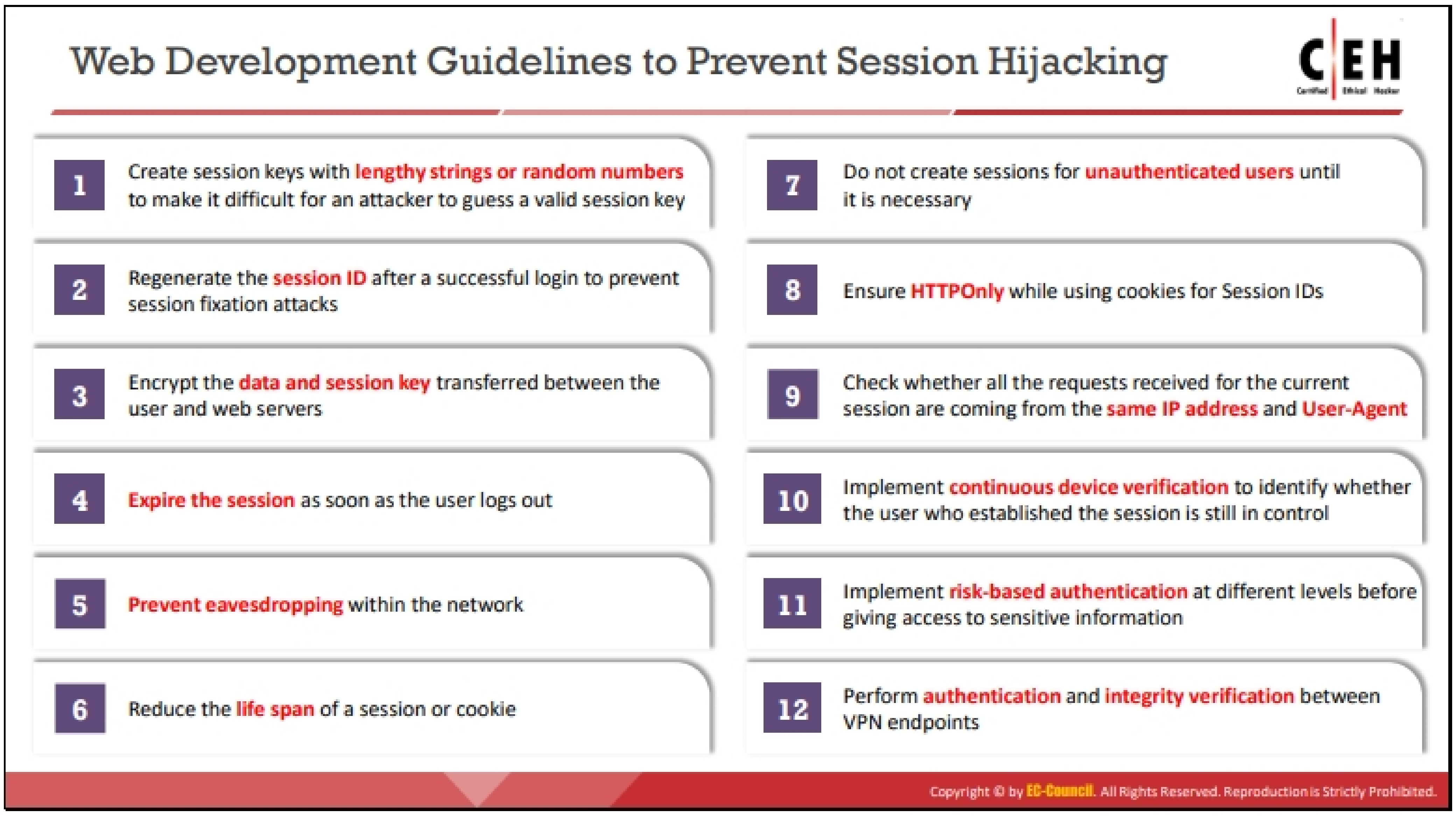
Protecting against Session Hijacking

1. Use **Secure Shell (SSH)** to create a secure communication channel
2. Implement the **log-out functionality** for the user to end the session
3. Generate the **session ID** after a successful login and accept only session IDs generated by the server only
4. Ensure that data in transit is **encrypted** and implement the **defense-in-depth** mechanism
5. Use **string** or a **long random number** as a session key
6. Use different **usernames** and **passwords** for different accounts
7. Implement **timeout()** to destroy the session when expired
8. Do not transport session ID in **query string**
9. Ensure that **client-side** and **server-side** protection software are active and up-to-date
10. Use **strong authentication** (e.g., Kerberos) or peer-to-peer virtual private networks (VPNs)
11. Configure the appropriate **internal** and **external spoof rules** on gateways
12. Use **IDS products** or **ARPwatch** for monitoring ARP cache poisoning
13. Use **HTTP Public Key Pinning** (HPKP) to allow users authenticate web servers
14. Enable browsers to **verify website authenticity** using network notary servers

Protecting against Session Hijacking

- Use the Secure Shell (SSH) to create a secure communication channel.
- Pass authentication cookies over HTTPS connections.
- Implement the log-out functionality for the user to end the session.
- Generate a session ID after a successful login and accept session IDs generated by the server only.
- Ensure that data in transit are encrypted and implement the defense-in-depth mechanism.
- Use strings or long random numbers as session keys.
- Use different usernames and passwords for different accounts.
- Educate employees and minimize remote access.
- Implement `timeout()` to destroy sessions when expired.
- Avoid including the session ID in the URL or query string.
- Use switches rather than hubs and limit incoming connections.
- Ensure client-side and server-side protection software are in the active state and up to date.
- Use strong authentication (such as Kerberos) or peer-to-peer virtual private networks (VPNs).
- Configure appropriate internal and external spoof rules on gateways.

- Use IDS products or ARPwatch for monitoring ARP cache poisoning.
- Use encrypted protocols available in the OpenSSH suite.
- Use firewalls and browser settings to confine cookies.
- Protect authentication cookies with SSL.
- Regularly update platform patches to fix TCP/IP vulnerabilities (e.g., predictable packet sequences).
- Use IPsec to encrypt session information.
- Use HTTP Public Key Pinning (HPKP) to allow users to authenticate web servers.
- Enable browsers to verify website authenticity using network notary servers.
- Implement DNS-based authentication of named entities.
- Disable compression mechanisms of HTTP requests.
- Use cipher-chaining block (CBC) ciphers incorporating random padding up to 255 bytes, thereby making the extraction of confidential information difficult for an attacker.
- Restrict the cross-site scripts known as cross-site request forgery (CSRF) from the client side.
- Upgrade web browsers to the latest versions.
- Use vulnerability scanners such as masscan to detect any insecure configuration of HTTPS session settings on sites.



Web Development Guidelines to Prevent Session Hijacking

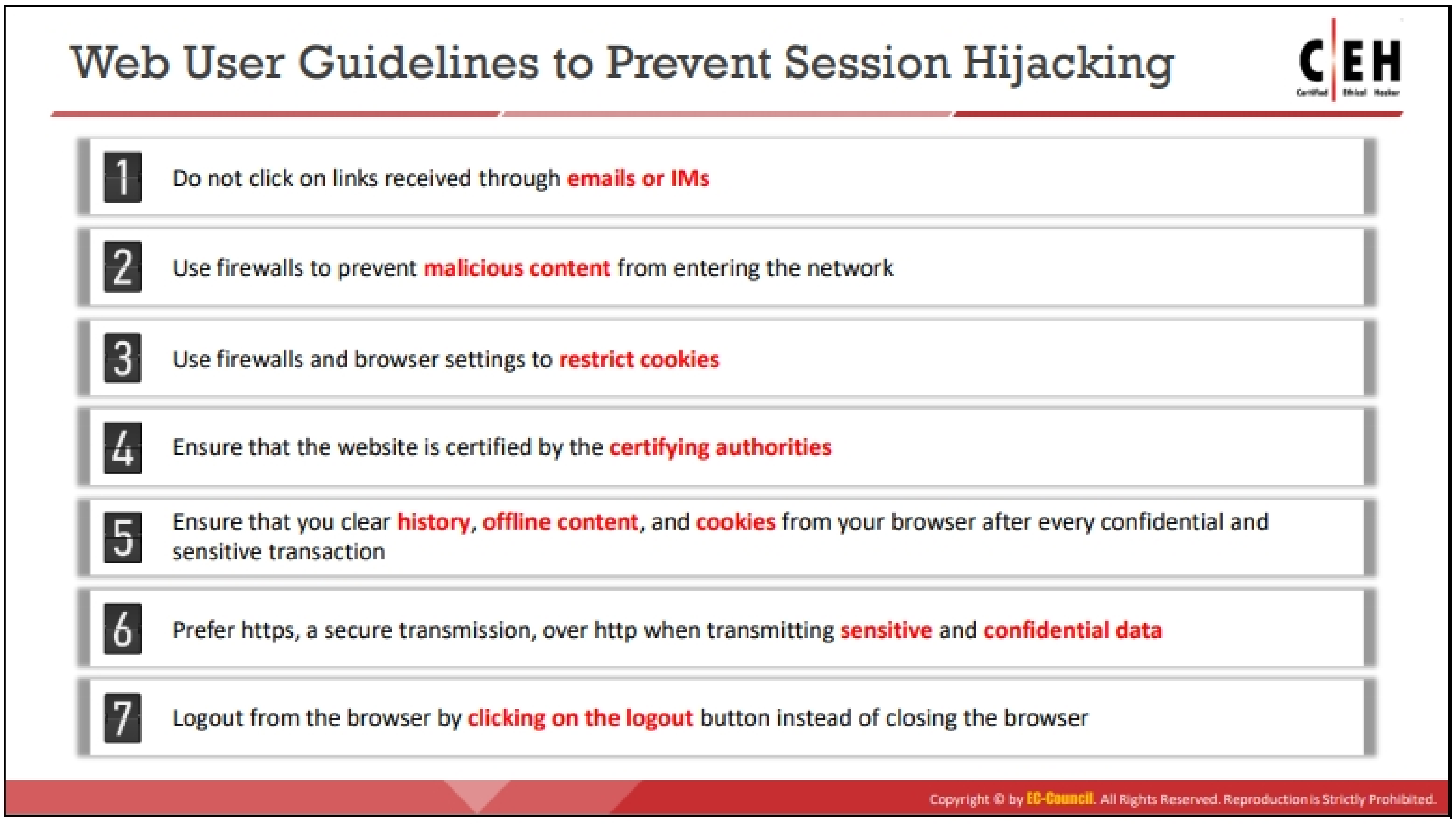
1. Create session keys with **lengthy strings or random numbers** to make it difficult for an attacker to guess a valid session key
2. Regenerate the **session ID** after a successful login to prevent session fixation attacks
3. Encrypt the **data and session key** transferred between the user and web servers
4. **Expire the session** as soon as the user logs out
5. **Prevent eavesdropping** within the network
6. Reduce the **life span** of a session or cookie
7. Do not create sessions for **unauthenticated users** until it is necessary
8. Ensure **HTTPOnly** while using cookies for Session IDs
9. Check whether all the requests received for the current session are coming from the **same IP address** and **User-Agent**
10. Implement **continuous device verification** to identify whether the user who established the session is still in control
11. Implement **risk-based authentication** at different levels before giving access to sensitive information
12. Perform **authentication** and **integrity verification** between VPN endpoints

Web Development Guidelines to Prevent Session Hijacking

An attacker usually hijacks a session by exploiting the vulnerabilities in mechanisms used for session establishment. Web developers often ignore security. During the development process, web developers should consider the following guidelines to minimize/eliminate the risk of session hijacking.

- Create session keys with lengthy strings or random numbers so that it is difficult for an attacker to guess a valid session key.
- Regenerate the session ID after a successful login to prevent session fixation attacks.
- Encrypt the data and session key transferred between the user and web servers.
- Implement the Secure Sockets Layer (SSL) to encrypt all the information in transit via the network.
- Make the session expire as soon as the user logs out.
- Prevent eavesdropping within the network.
- Reduce the life span of a session or cookie.
- Use restrictive cache directives for all the web traffic through HTTP and HTTPS, such as the "**Cache-Control: no-cache, no-store**" and "**Pragma: no-cache**" HTTP headers and/or equivalent META tags on all or (at least) sensitive web pages.
- Do not create sessions for unauthenticated users unless necessary.
- Ensure HTTPOnly while using cookies for session IDs.
- Use a secure flag to send cookies in HTTPS requests and encrypt them before sending across the network.

- Check whether all the requests received for the current session originate from the same IP address and user agent.
- Implement continuous device verification to identify whether the user who established the session is still in control.
- Implement risk-based authentication at different levels before granting access to sensitive information.
- Perform authentication and integrity verification between VPN endpoints.



Web User Guidelines to Prevent Session Hijacking

1 Do not click on links received through **emails or IMs**

2 Use firewalls to prevent **malicious content** from entering the network

3 Use firewalls and browser settings to **restrict cookies**

4 Ensure that the website is certified by the **certifying authorities**

5 Ensure that you clear **history**, **offline content**, and **cookies** from your browser after every confidential and sensitive transaction

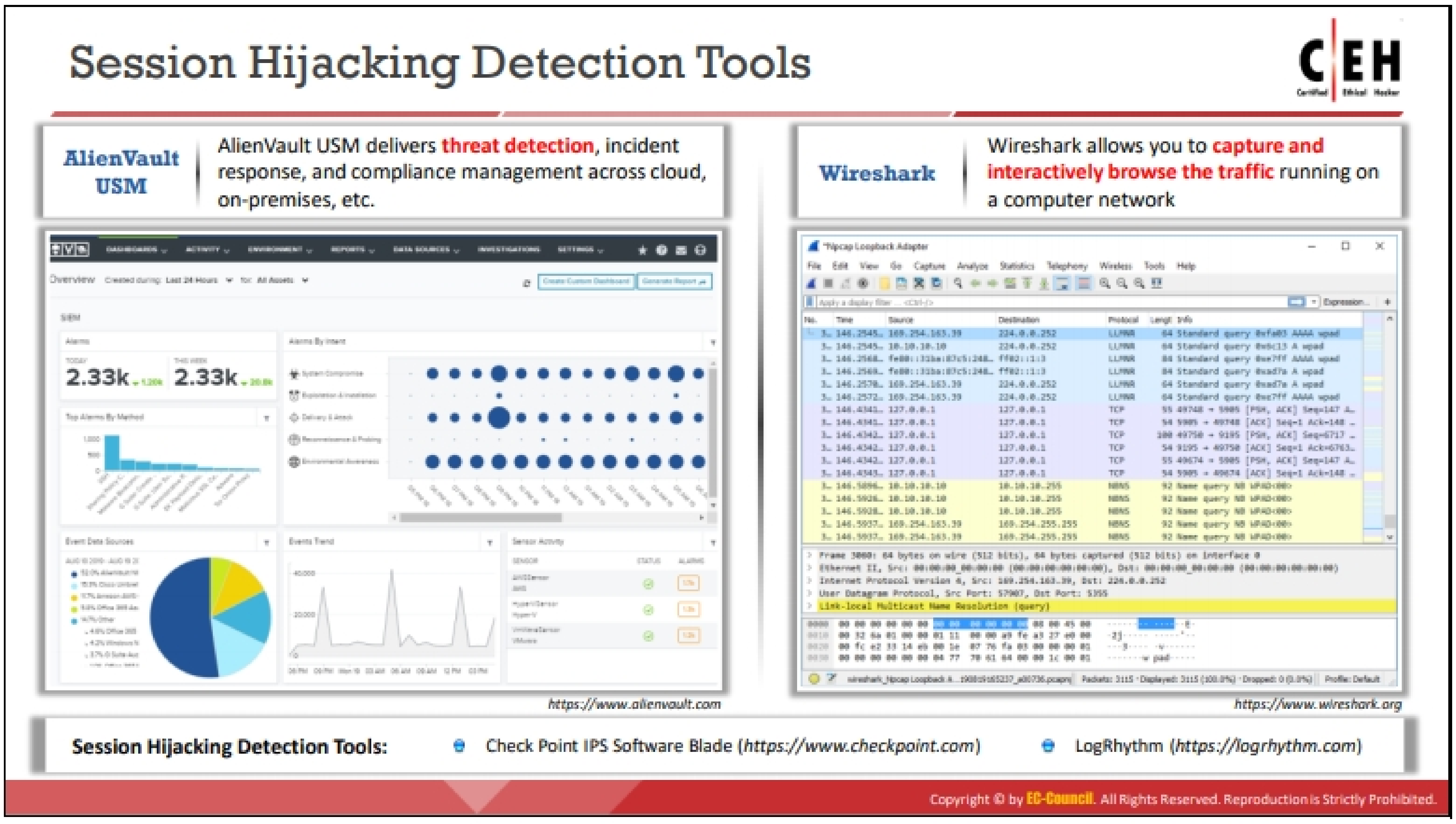
6 Prefer https, a secure transmission, over http when transmitting **sensitive** and **confidential data**

7 Logout from the browser by **clicking on the logout** button instead of closing the browser

Web User Guidelines to Prevent Session Hijacking

The following are some guidelines for web users to defend against session hijacking.

- Do not click on links received through emails or instant messages (IMs).
- Use firewalls to prevent malicious content from entering the network.
- Use firewalls and browser settings to restrict cookies.
- Ensure that the website is certified by appropriate certifying authorities.
- Ensure that the history, offline content, and cookies are cleared from the browser after every confidential and sensitive transaction.
- Give preference to HTTPS, a secure transmission protocol, over HTTP when transmitting sensitive and confidential data.
- Logout from the browser by clicking on the logout button instead of closing the browser.
- Verify and disable add-ons from untrusted sites. Enable add-ons only if necessary.
- Practice using a one-time password for critical data transactions (e.g., credit card transactions).
- Frequently update anti-virus signatures to prevent the automatic installation of malware that attempts to steal cookies.



Session Hijacking Detection Tools

AlienVault USM AlienVault USM delivers threat detection, incident response, and compliance management across cloud, on-premises, etc.

https://www.alienvault.com

Wireshark Wireshark allows you to capture and interactively browse the traffic running on a computer network

https://www.wireshark.org

Session Hijacking Detection Tools:

- Check Point IPS Software Blade (*https://www.checkpoint.com*)
- LogRhythm (*https://logrhythm.com*)

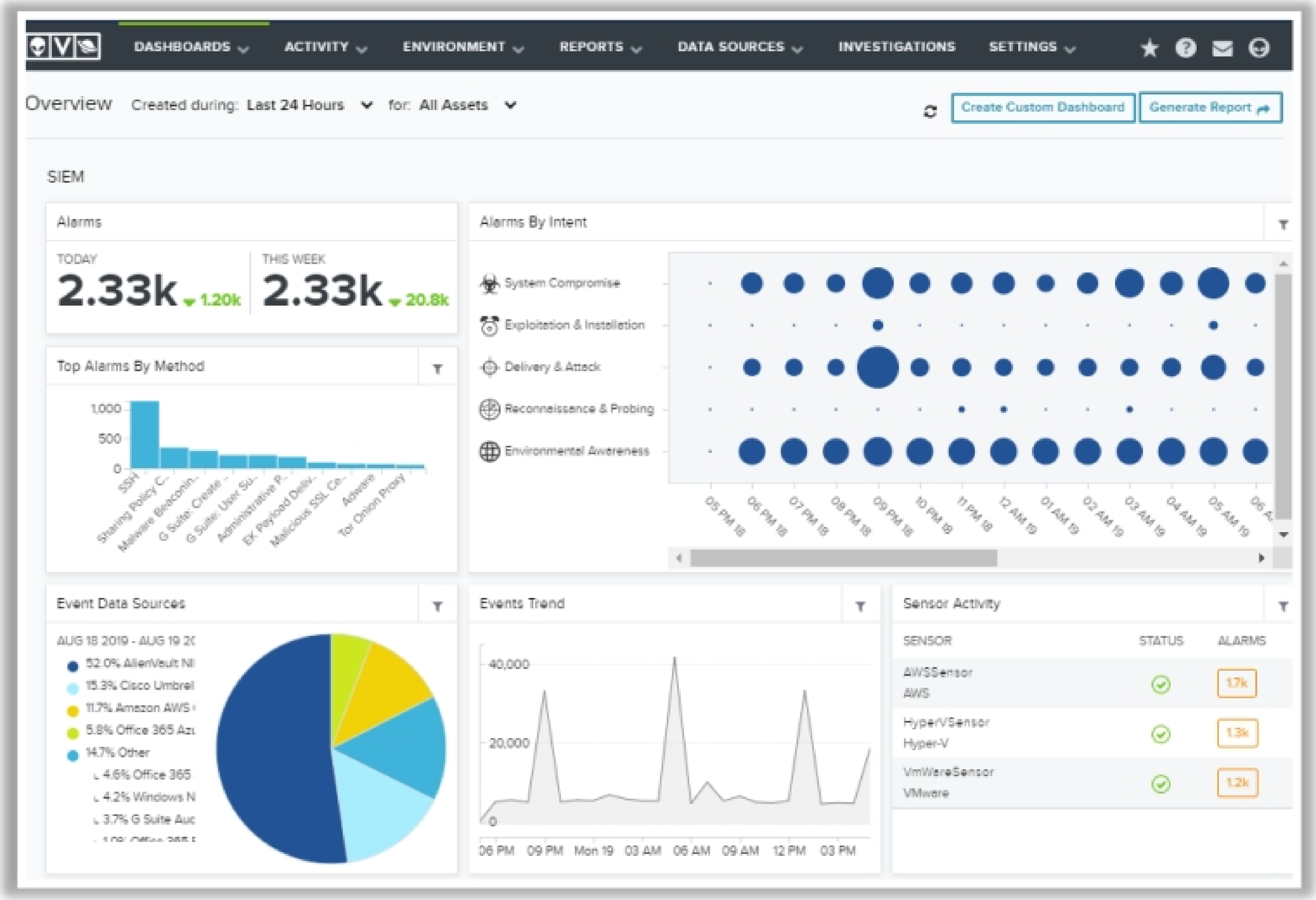
Session Hijacking Detection Tools

Session hijacking attacks are difficult to detect, and in most cases, attacks go unnoticed, causing severe leakage of confidential data. Tools such as packet sniffers, IDSs, and security information and event management (SIEM) can be used to detect session hijacking attacks.

- **AlientVault USM**

  Source: *https://www.alienvault.com*

  AlienVault Unified Security Management (USM) offers powerful threat detection, incident response, and compliance management across cloud, on-premises, and hybrid environments. Security professionals can use this tool for detecting session hijacking attempts and perform asset discovery, intrusion detection, security automation, SIEM and log management, endpoint detection and response, threat detection, threat intelligence, and vulnerability assessment.


DASHBOARDS
ACTIVITY
ENVIRONMENT
REPORTS
DATA SOURCES
INVESTIGATIONS
SETTINGS
Overview
Created during: Last 24 Hours
for: All Assets
Create Custom Dashboard
Generate Report
SIEM
Alarms
TODAY
2.33k 1.20k
THIS WEEK
2.33k 20.8k
Alarms By Intent
System Compromise
Exploitation & Installation
Delivery & Attack
Reconnaissance & Probing
Environmental Awareness
05 PM 18
06 PM 18
07 PM 18
08 PM 18
09 PM 18
10 PM 18
11 PM 18
12 AM 19
01 AM 19
02 AM 19
03 AM 19
04 AM 19
05 AM 19
Top Alarms By Method
1,000
500
0
SSH
Sharing Policy C...
Malware Beaconin...
G Suite: Create ...
G Suite: User Su...
Administrative P...
EK Payload Deliv...
Malicious SSL Ce...
Adware
Tor Onion Proxy
Event Data Sources
AUG 18 2019 - AUG 19 20
52.0% AlienVault NI
15.3% Cisco Umbrel
11.7% Amazon AWS
5.8% Office 365 Azu
14.7% Other
4.6% Office 365
4.2% Windows N
3.7% G Suite Aud
Events Trend
40,000
20,000
0
06 PM
09 PM
Mon 19
03 AM
06 AM
09 AM
12 PM
03 PM
Sensor Activity

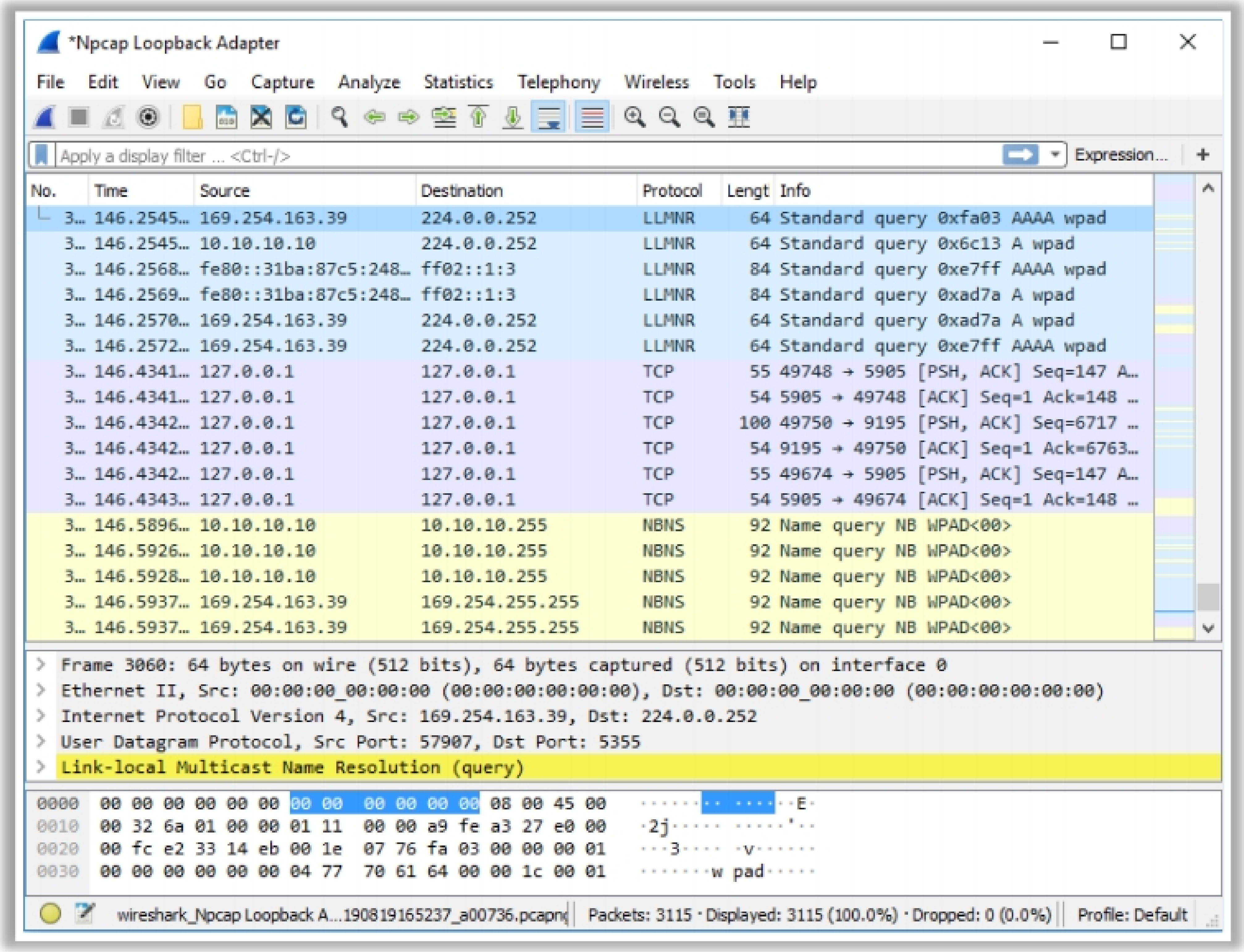
SENSOR	STATUS	ALARMS
AWSSensor AWS	✓	1.7k
HyperVSensor Hyper-V	✓	1.3k
VmWareSensor VMware	✓	1.2k

Figure 11.31: Screenshot of AlientVault USM

- **Wireshark**

  Source: *https://www.wireshark.org*

  Wireshark allows users to capture and interactively browse the traffic on a network. This tool uses Winpcap to capture packets. Therefore, it can only capture packets on the networks supported by Winpcap. It captures live network traffic from Ethernet, IEEE 802.11, Point-to-Point Protocol/High-level Data Link Control (PPP/HDLC), Asynchronous Transfer Mode (ATM), Bluetooth, Universal Serial Bus (USB), Token Ring, Frame Relay, and Fiber Distributed Data Interface (FDDI) networks. Security professionals use Wireshark to monitor and detect session hijacking attempts.

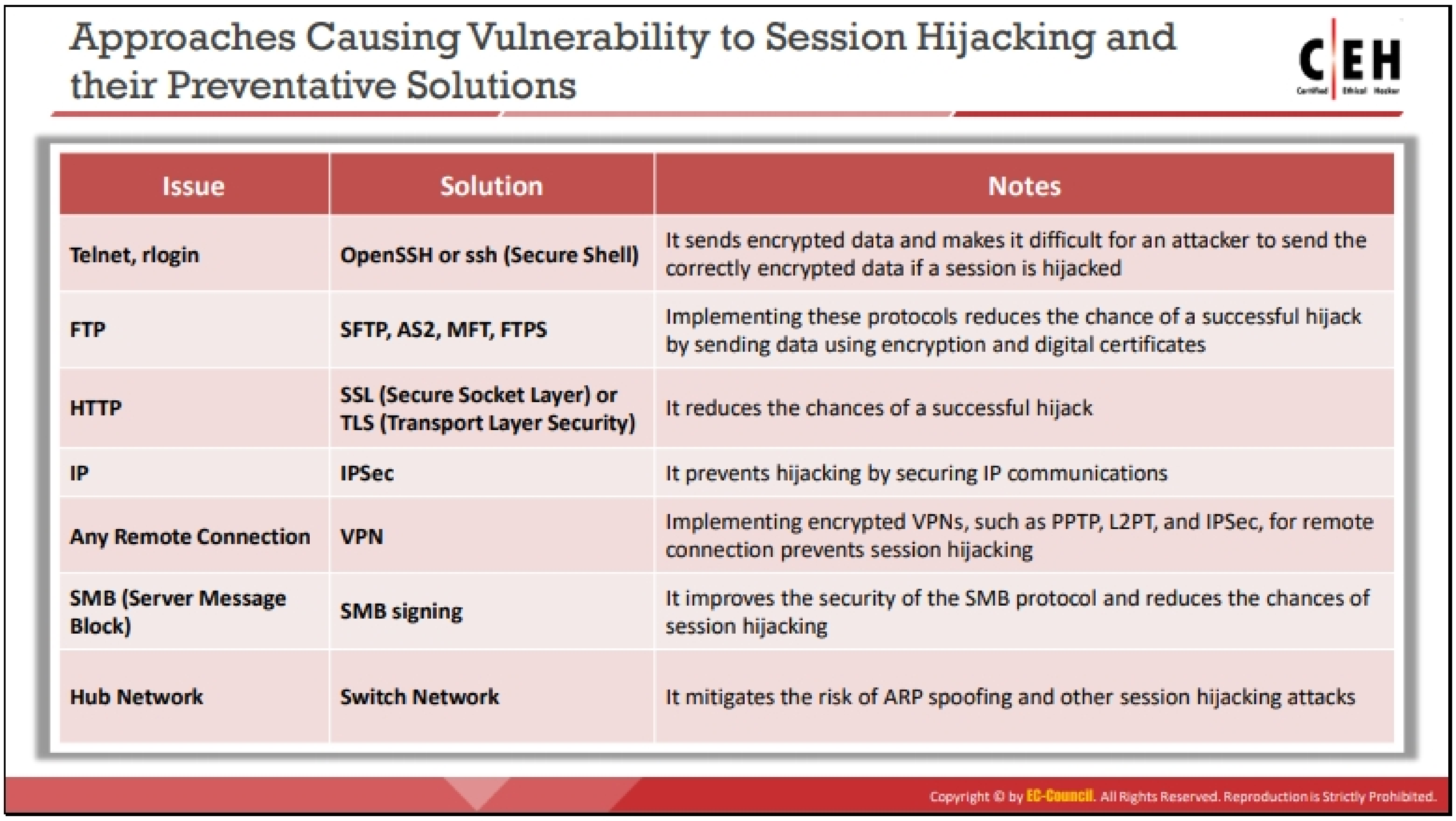


```
No.   Time        Source                   Destination      Protocol  Lengt Info
 3…   146.2545…   169.254.163.39           224.0.0.252      LLMNR      64 Standard query 0xfa03 AAAA wpad
 3…   146.2545…   10.10.10.10              224.0.0.252      LLMNR      64 Standard query 0x6c13 A wpad
 3…   146.2568…   fe80::31ba:87c5:248…     ff02::1:3        LLMNR      84 Standard query 0xe7ff AAAA wpad
 3…   146.2569…   fe80::31ba:87c5:248…     ff02::1:3        LLMNR      84 Standard query 0xad7a A wpad
 3…   146.2570…   169.254.163.39           224.0.0.252      LLMNR      64 Standard query 0xad7a A wpad
 3…   146.2572…   169.254.163.39           224.0.0.252      LLMNR      64 Standard query 0xe7ff AAAA wpad
 3…   146.4341…   127.0.0.1                127.0.0.1        TCP        55 49748 → 5905 [PSH, ACK] Seq=147 A…
 3…   146.4341…   127.0.0.1                127.0.0.1        TCP        54 5905 → 49748 [ACK] Seq=1 Ack=148 …
 3…   146.4342…   127.0.0.1                127.0.0.1        TCP       100 49750 → 9195 [PSH, ACK] Seq=6717 …
 3…   146.4342…   127.0.0.1                127.0.0.1        TCP        54 9195 → 49750 [ACK] Seq=1 Ack=6763…
 3…   146.4342…   127.0.0.1                127.0.0.1        TCP        55 49674 → 5905 [PSH, ACK] Seq=147 A…
 3…   146.4343…   127.0.0.1                127.0.0.1        TCP        54 5905 → 49674 [ACK] Seq=1 Ack=148 …
 3…   146.5896…   10.10.10.10              10.10.10.255     NBNS       92 Name query NB WPAD<00>
 3…   146.5926…   10.10.10.10              10.10.10.255     NBNS       92 Name query NB WPAD<00>
 3…   146.5928…   10.10.10.10              10.10.10.255     NBNS       92 Name query NB WPAD<00>
 3…   146.5937…   169.254.163.39           169.254.255.255  NBNS       92 Name query NB WPAD<00>
 3…   146.5937…   169.254.163.39           169.254.255.255  NBNS       92 Name query NB WPAD<00>

> Frame 3060: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0
> Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
> Internet Protocol Version 4, Src: 169.254.163.39, Dst: 224.0.0.252
> User Datagram Protocol, Src Port: 57907, Dst Port: 5355
> Link-local Multicast Name Resolution (query)

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  00 32 6a 01 00 00 01 11  00 00 a9 fe a3 27 e0 00   ·2j····· ·····'··
0020  00 fc e2 33 14 eb 00 1e  07 76 fa 03 00 00 00 01   ···3···· ·v······
0030  00 00 00 00 00 00 04 77  70 61 64 00 00 1c 00 01   ·······w pad·····

wireshark_Npcap Loopback A…190819165237_a00736.pcapng | Packets: 3115 · Displayed: 3115 (100.0%) · Dropped: 0 (0.0%) | Profile: Default
```

Figure 11.32: Screenshot of Wireshark

The following are some additional session hijacking detection tools:

- Check Point IPS Software Blade (*https://www.checkpoint.com*)
- LogRhythm (*https://logrhythm.com*)
- SolarWinds Log & Event Manager (*https://www.solarwinds.com*)
- IBM Security Network Intrusion Prevention System (*https://www.ibm.com*)



Approaches Causing Vulnerability to Session Hijacking and their Preventative Solutions

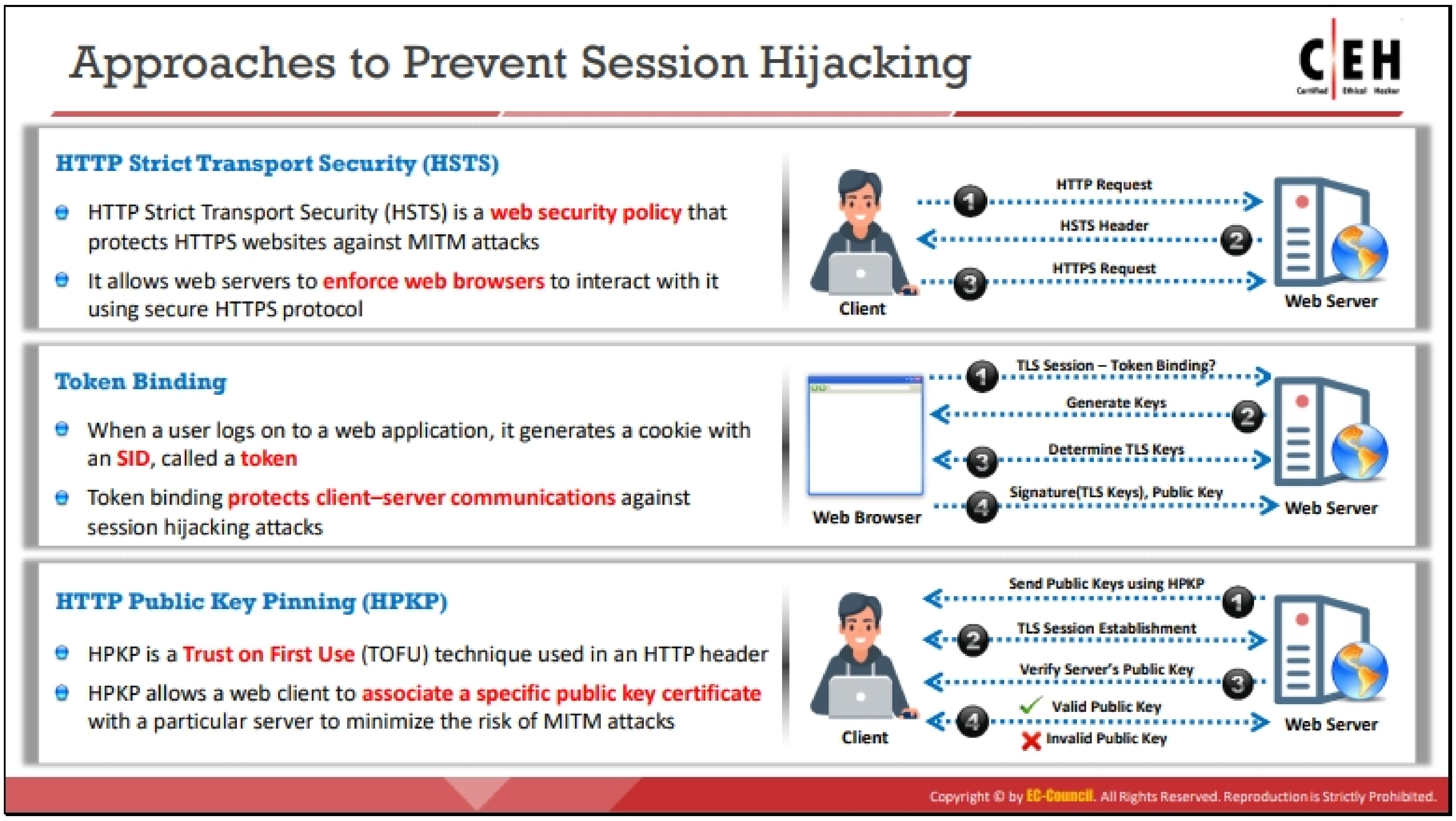
Issue	Solution	Notes
Telnet, rlogin	OpenSSH or ssh (Secure Shell)	It sends encrypted data and makes it difficult for an attacker to send the correctly encrypted data if a session is hijacked
FTP	SFTP, AS2, MFT, FTPS	Implementing these protocols reduces the chance of a successful hijack by sending data using encryption and digital certificates
HTTP	SSL (Secure Socket Layer) or TLS (Transport Layer Security)	It reduces the chances of a successful hijack
IP	IPSec	It prevents hijacking by securing IP communications
Any Remote Connection	VPN	Implementing encrypted VPNs, such as PPTP, L2PT, and IPSec, for remote connection prevents session hijacking
SMB (Server Message Block)	SMB signing	It improves the security of the SMB protocol and reduces the chances of session hijacking
Hub Network	Switch Network	It mitigates the risk of ARP spoofing and other session hijacking attacks

Approaches Causing Vulnerability to Session Hijacking and their Preventative Solutions

Implementing encryption and signing protocols prevents attackers from hijacking sessions. Below table lists various issues and their respective solutions that, upon implementation, prevent or impede the hijacking of a valid session.

Issue	Solution	Notes
Telnet, rlogin	Secure Shell (SSH) or OpenSSH	It sends encrypted data and makes it difficult for an attacker to send correctly encrypted data if a session is hijacked.
FTP	SSH FTP (SFTP), Applicability Statement 2 (AS2), managed file transfer (MFT), FTP Secure (FTPS)	Implementing these protocols reduces the chance of a successful hijack by sending data using encryption and digital certificates.
HTTP	Secure Sockets Layer (SSL) or Transport Layer Security (TLS)	It reduces the chances of a successful hijack.
IP	IPsec	It prevents hijacking by securing IP communications.
Any Remote Connection	Virtual private network (VPN)	Implementing encrypted VPNs such as PPTP, Layer 2 Protocol Tunneling (L2PT), and IPsec, for remote connections prevents session hijacking.
Server Message Block (SMB)	SMB signing	It improves the security of the SMB protocol and reduces the chances of session hijacking.
Hub Network	Switch network	It mitigates the risk of ARP spoofing and other session hijacking attacks.

Table 11.1: Approaches causing vulnerability to session hijacking and their preventative solutions



Approaches to Prevent Session Hijacking

HTTP Strict Transport Security (HSTS)

- HTTP Strict Transport Security (HSTS) is a **web security policy** that protects HTTPS websites against MITM attacks
- It allows web servers to **enforce web browsers** to interact with it using secure HTTPS protocol

HTTP Request
HSTS Header
HTTPS Request
Client
Web Server

Token Binding

- When a user logs on to a web application, it generates a cookie with an **SID**, called a **token**
- Token binding **protects client–server communications** against session hijacking attacks

TLS Session – Token Binding?
Generate Keys
Determine TLS Keys
Signature(TLS Keys), Public Key
Web Browser
Web Server

HTTP Public Key Pinning (HPKP)

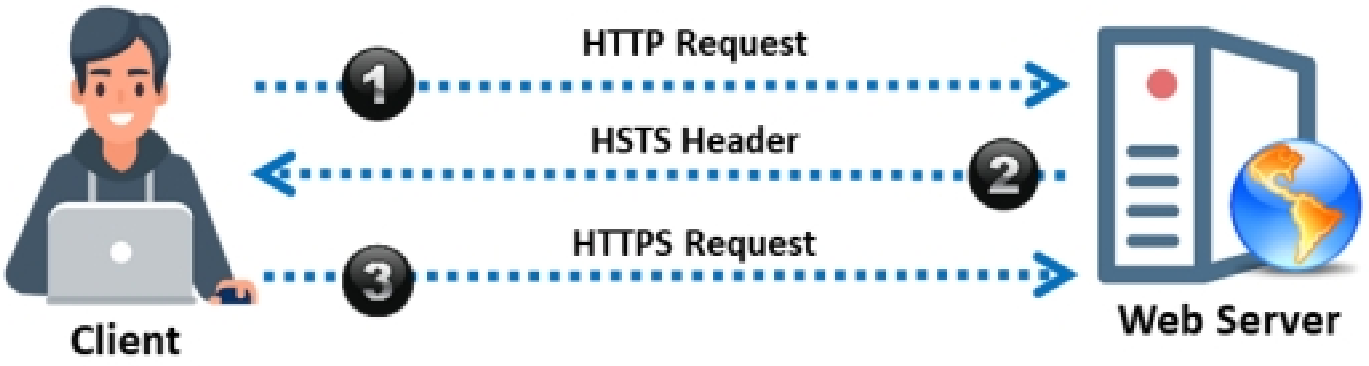
- HPKP is a **Trust on First Use** (TOFU) technique used in an HTTP header
- HPKP allows a web client to **associate a specific public key certificate** with a particular server to minimize the risk of MITM attacks

Send Public Keys using HPKP
TLS Session Establishment
Verify Server's Public Key
✓ Valid Public Key
✗ Invalid Public Key
Client
Web Server

Approaches to Prevent Session Hijacking

- **HTTP Strict Transport Security (HSTS)**

  HTTP Strict Transport Security (HSTS) is a web security policy that protects HTTPS websites against MITM attacks. The HSTS policy helps web servers force web browsers to interact with them using HTTPS. With the HSTS policy, all insecure HTTP connections are automatically converted into HTTPS connections. This policy ensures that all the communication between a web server and web browser is encrypted and that all responses that are delivered and received originate from an authenticated server.



HTTP Request
HSTS Header
HTTPS Request
Client
Web Server

Figure 11.33: Implementation of HSTS

- **Token Binding**

  When a user logs into a web application, a cookie with a session ID, called a token, is generated. The user utilizes this random token to send requests to the server and access resources. An attacker can impersonate the user and hijack the connection by capturing and reusing a valid session ID. Token binding protects client–server communication against session hijacking attacks. The client creates a public–private key pair for every connection to a remote server. When a client connects to the server, it generates a signature using a private key and sends this signature along with its public key to the server. The server verifies the signature using the client's public key. This ensures that the message was sent by an authentic client because only the client has its private key. Even if an attacker captures the signature, it is not possible for them to regenerate the signature or reuse it for another connection. For every new connection, a new pair of public and private keys are used.

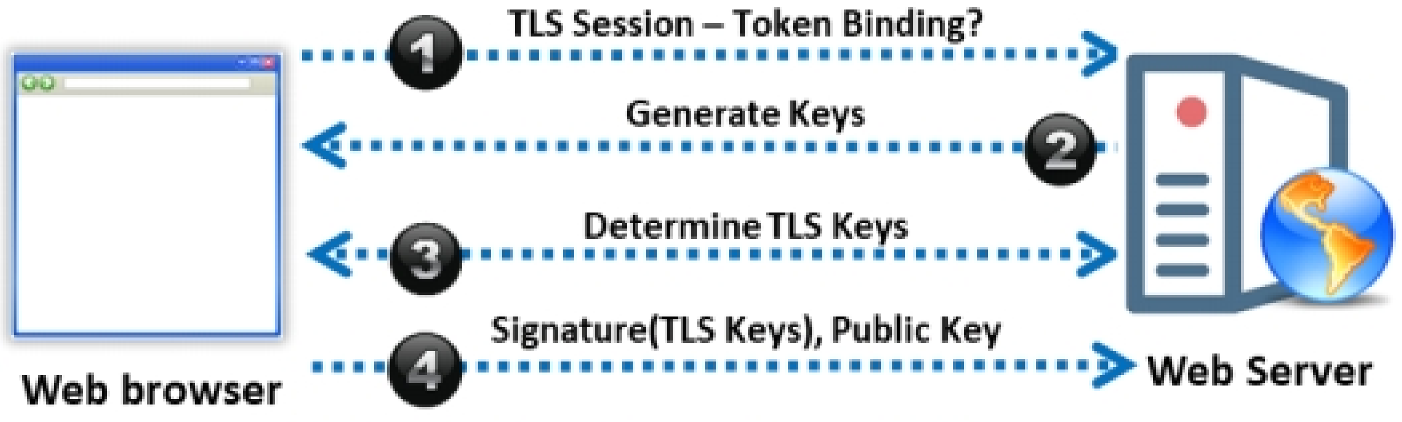

1 TLS Session – Token Binding?
2 Generate Keys
3 Determine TLS Keys
4 Signature(TLS Keys), Public Key
Web browser
Web Server

Figure 11.34: Implementation of token binding

- **HTTP Public Key Pinning (HPKP)**

  HTTP Public Key Pinning (HPKP) is a trust on first use (TOFU) technique used in an HTTP header that allows a web client to associate a specific public key certificate with a particular server to minimize the risk of MITM attacks based on fraudulent certificates. In TLS sessions, to verify the authenticity of a server's public key, the public key is enclosed in an X.509 digital certificate, which is signed by a certification authority (CA). By compromising any CA, attackers can perform MITM attacks on various TLS sessions. HPKP protects TLS sessions from such attacks by delivering to the client the list of public keys owned by a web server. When the client connects to the web server, it verifies the server's certificate in the certificate chain obtained using HPKP. If the server sends any unidentified public key, the client issues a warning message to the user.

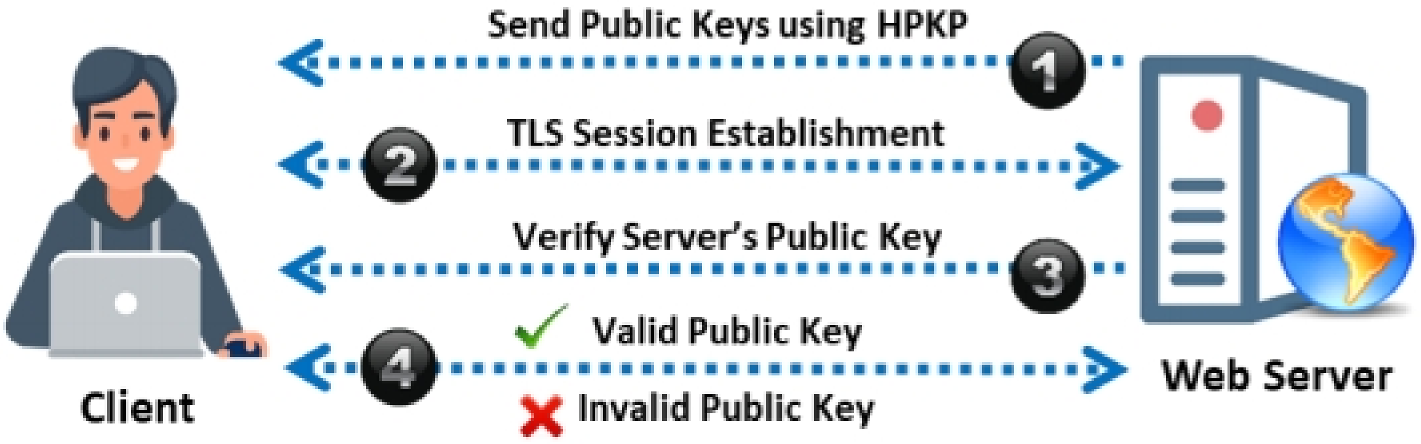
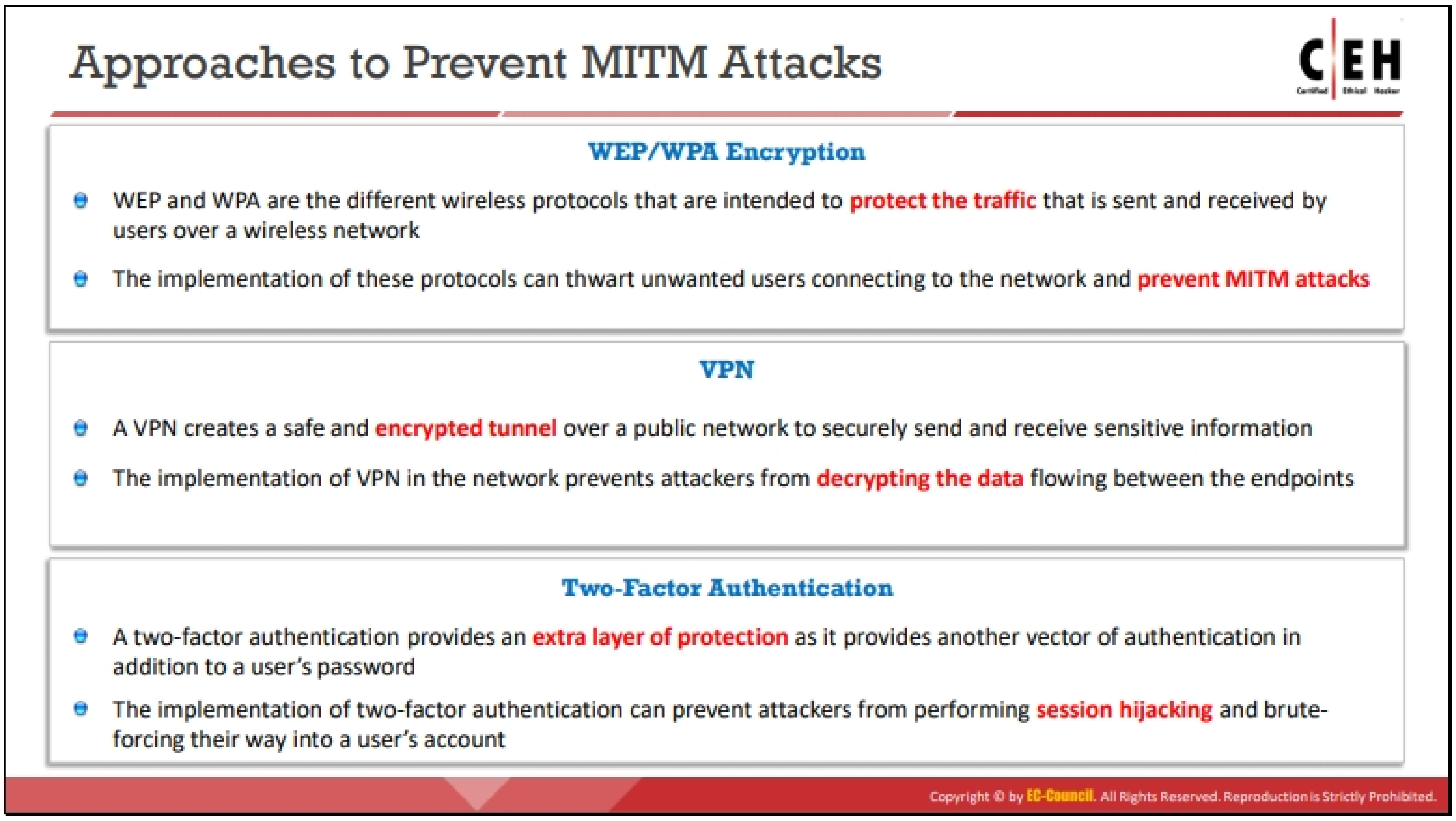

1 Send Public Keys using HPKP
2 TLS Session Establishment
3 Verify Server's Public Key
4
✓ Valid Public Key
✗ Invalid Public Key
Client
Web Server

Figure 11.35: Implementation of HPKP



Approaches to Prevent MITM Attacks

WEP/WPA Encryption

- WEP and WPA are the different wireless protocols that are intended to **protect the traffic** that is sent and received by users over a wireless network
- The implementation of these protocols can thwart unwanted users connecting to the network and **prevent MITM attacks**

VPN

- A VPN creates a safe and **encrypted tunnel** over a public network to securely send and receive sensitive information
- The implementation of VPN in the network prevents attackers from **decrypting the data** flowing between the endpoints

Two-Factor Authentication

- A two-factor authentication provides an **extra layer of protection** as it provides another vector of authentication in addition to a user's password
- The implementation of two-factor authentication can prevent attackers from performing **session hijacking** and brute-forcing their way into a user's account

Approaches to Prevent MITM Attacks

Man-in-the-middle (MITM) attacks are the most common attacks in which the attackers can intercept the traffic between two endpoints. The victim may not realize the effect of this attack, because it is mostly passive in nature. Because the detection of MITM attacks is difficult, they can only be prevented using various measures. The following are some approaches to prevent MITM attacks:

- **WEP/WPA Encryption**

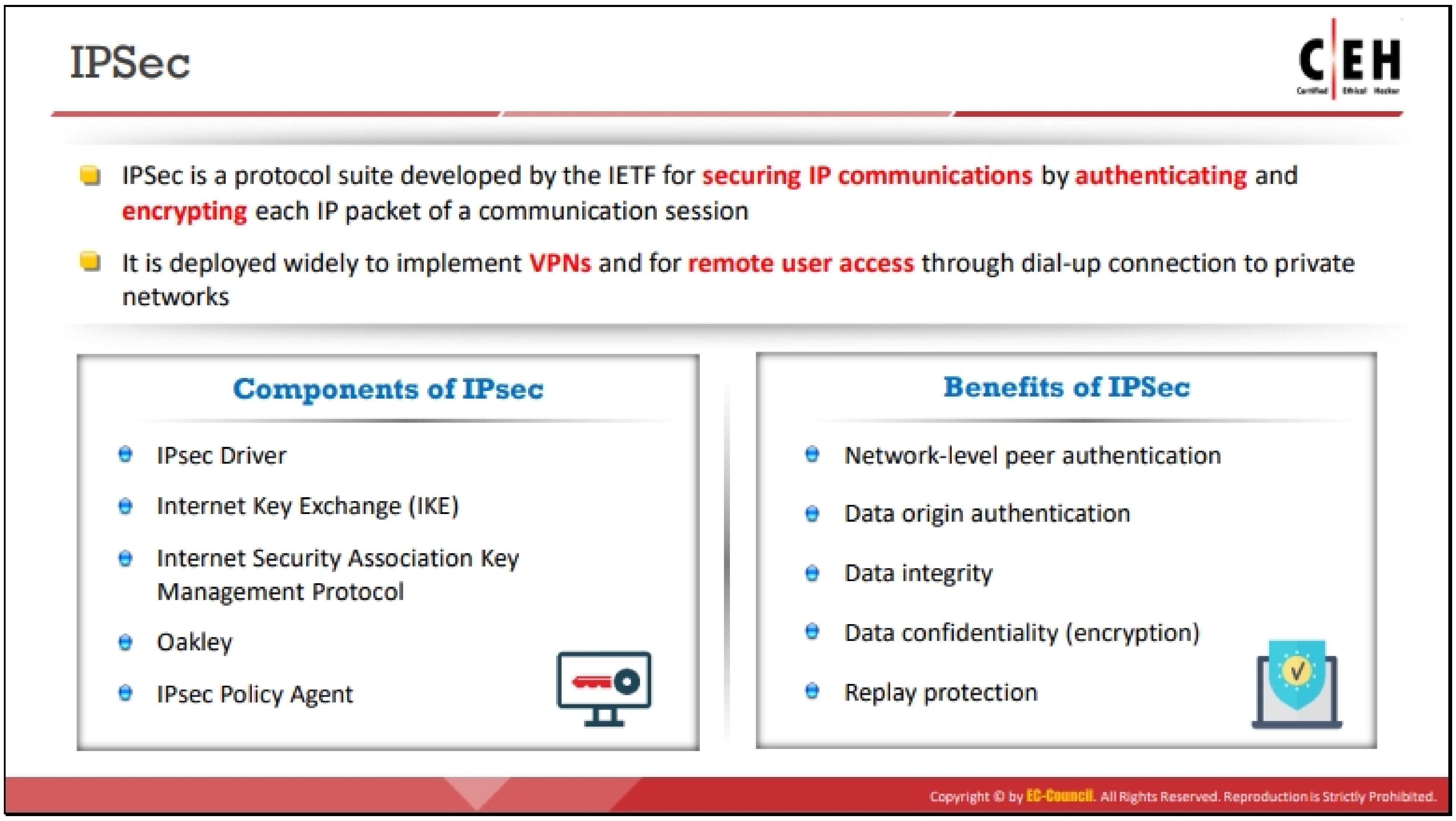
  Wired Equivalent Privacy (WEP) and Wireless Protected Access (WPA) are wireless protocols that are intended to protect the traffic that is sent and received by users over a wireless network. The implementation of these protocols can thwart the attempts of unwanted users to connect to the network. A weak encryption mechanism enables attackers to brute force credentials and enter the target network to perform an MITM attack.

- **VPN**

  A VPN creates a safe and encrypted tunnel over a public network to securely send and receive sensitive information. It creates a subnet by using key-based encryption for secure communication between endpoints. The implementation of a VPN in the network prevents attackers from decrypting the data flowing between the endpoints.

- **Two-Factor Authentication**

  Two-factor authentication provides an extra layer of protection because it serves as a vector of authentication in addition to a user's password. Therefore, the implementation of two-factor authentication can prevent attackers from performing session hijacking and brute forcing to compromise a user's account.



IPSec

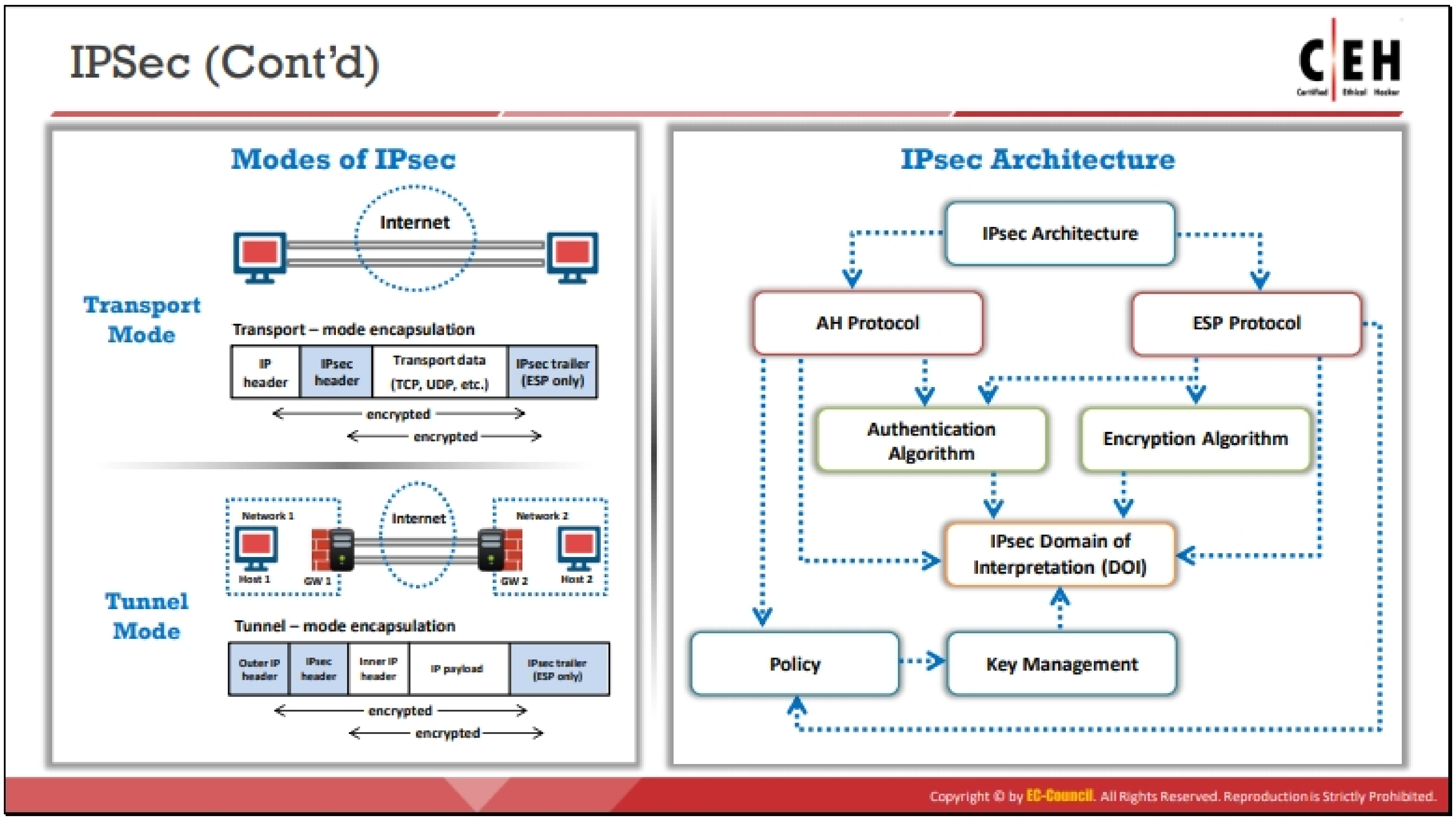
- IPSec is a protocol suite developed by the IETF for **securing IP communications** by **authenticating** and **encrypting** each IP packet of a communication session
- It is deployed widely to implement **VPNs** and for **remote user access** through dial-up connection to private networks

Components of IPsec

- IPsec Driver
- Internet Key Exchange (IKE)
- Internet Security Association Key Management Protocol
- Oakley
- IPsec Policy Agent

Benefits of IPSec

- Network-level peer authentication
- Data origin authentication
- Data integrity
- Data confidentiality (encryption)
- Replay protection



IPSec (Cont'd)

Modes of IPsec

Transport Mode

Transport – mode encapsulation

Internet
IP header
IPsec header
Transport data (TCP, UDP, etc.)
IPsec trailer (ESP only)
encrypted
encrypted

Tunnel Mode

Tunnel – mode encapsulation

Network 1
Internet
Network 2
Host 1
GW 1
GW 2
Host 2
Outer IP header
IPsec header
Inner IP header
IP payload
IPsec trailer (ESP only)
encrypted
encrypted

IPsec Architecture

IPsec Architecture
AH Protocol
ESP Protocol
Authentication Algorithm
Encryption Algorithm
IPsec Domain of Interpretation (DOI)
Policy
Key Management

IPsec

Internet Protocol Security (IPsec) is a set of protocols that the Internet Engineering Task Force (IETF) developed to support the secure exchange of packets at the IP layer. It ensures interoperable cryptographically based security for IPv4 and IPv6, and it supports network level peer authentication, data origin authentication, data integrity, data confidentiality (encryption), and replay protection. It is widely used to implement VPNs and for remote user access through dial-up connection to private networks. It supports transport and tunnel encryption modes, although sending and receiving devices must share a public key.

IPsec policies can be assigned through the Group Policy configuration of Active Directory domains, organizational units, and IPsec deployment policies at the domain, site, or organizational-unit level. The security services offered by IPsec include the following:

- Rejection of replayed packets (a form of partial sequence integrity)
- Data confidentiality (encryption)
- Access control
- Connectionless integrity
- Data origin authentication
- Data integrity
- Limited traffic-flow confidentiality
- Network level peer authentication
- Replay protection

At the IP layer, IPsec provides all the above-mentioned services, offering the protection of IP and/or upper-layer protocols such as TCP, UDP, ICMP, and Border Gateway Protocol (BGP).

Components of IPsec

- **IPsec driver**: Software that performs protocol-level functions required to encrypt and decrypt packets.
- **Internet Key Exchange (IKE)**: An protocol that produces security keys for IPsec and other protocols.
- **Internet Security Association and Key Management Protocol (ISAKMP)**: Software that allows two computers to communicate by encrypting the data exchanged between them.
- **Oakley**: A protocol that uses the Diffie–Hellman algorithm to create a master key and a key that is specific to each session in IPsec data transfer.
- **IPsec Policy Agent**: A service included in Windows OS that enforces IPsec policies for all the network communications initiated from that system.

The following are the steps involved in the IPsec process.

- A consumer sends a message to a service provider.
- The consumer's IPsec driver attempts to match the outgoing packet's address or the packet type against the IP filter.
- The IPsec driver notifies ISAKMP to initiate security negotiations with the service provider.
- The service provider's ISAKMP receives the security negotiation request.

- Both principles initiate a key exchange, establishing an ISAKMP Security Association (SA) and a shared secret key.
- Both principles discuss the security level for the information exchange, establishing both IPsec SAs and keys.
- The consumer's IPsec driver transfers packets to the appropriate connection type for transmission to the service provider.
- The provider receives the packets and transfers them to the IPsec driver.
- The provider's IPsec uses the inbound SA and key to check the digital signature and begin decryption.
- The provider's IPsec driver transfers decrypted packets to the OSI transport layer for further processing.

Modes of IPsec

The configuration of IPsec involves two different modes: the tunnel mode and transport mode. These modes are associated with the functions of two core protocols: the Encapsulation Security Payload (ESP) and Authentication Header (AH). The model selection depends on the requirements and implementation of IPsec.

- **Transport Mode**

  In the transport mode (also ESP), IPsec encrypts only the payload of the IP packet, leaving the header untouched. It authenticates two connected computers and provides the option of encrypting data transfer. It is compatible with network address translation (NAT); therefore, it can be used to provide VPN services for networks utilizing NAT.

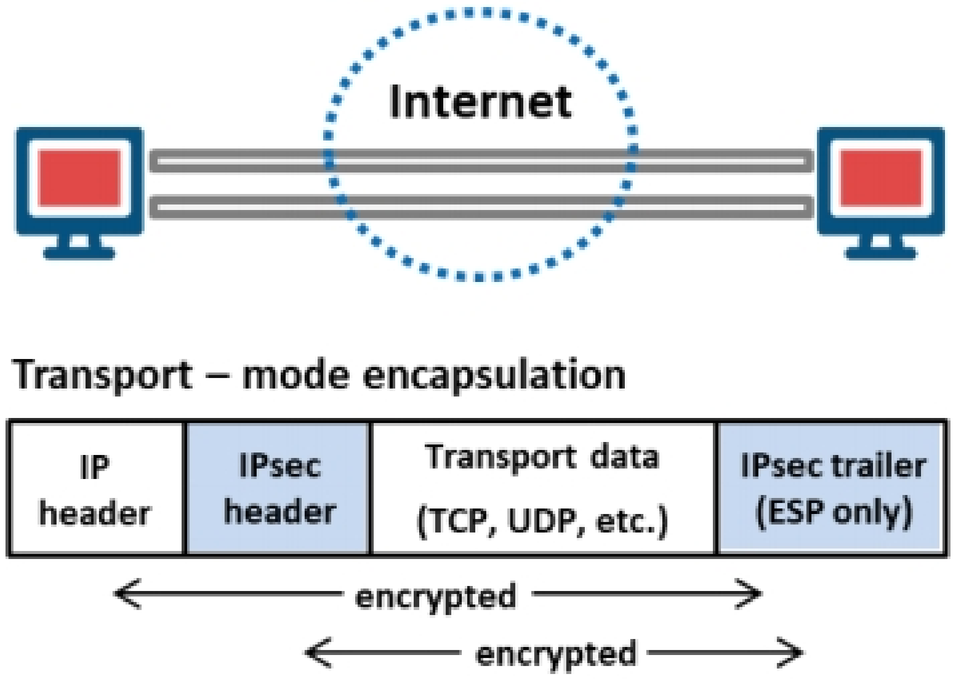


Figure 11.36: Transport mode encapsulation

- **Tunnel Mode**

  In the tunnel mode (also AH), the IPsec encrypts both the payload and header. Hence, in the tunnel mode has higher security than the transport mode. After receiving the data, the IPsec-compliant device performs decryption. The tunnel model is used to create VPNs over the Internet for network-to-network communication (e.g., between routers and link sites), host-to-network communication (e.g., remote user access), and host-to-host communication (e.g., private chat). It is compatible with NAT and supports NAT traversal.

In the tunnel mode, the system encrypts entire IP packets (payload and IP header) and encapsulates the encrypted packets into a new IP packet with a new header. In this mode, ESP encrypts and optionally authenticates entire inner IP packets, whereas AH authenticates entire inner IP packets and selected fields of outer IP headers. The tunnel mode is usually useful between two gateways or between a host and gateway.

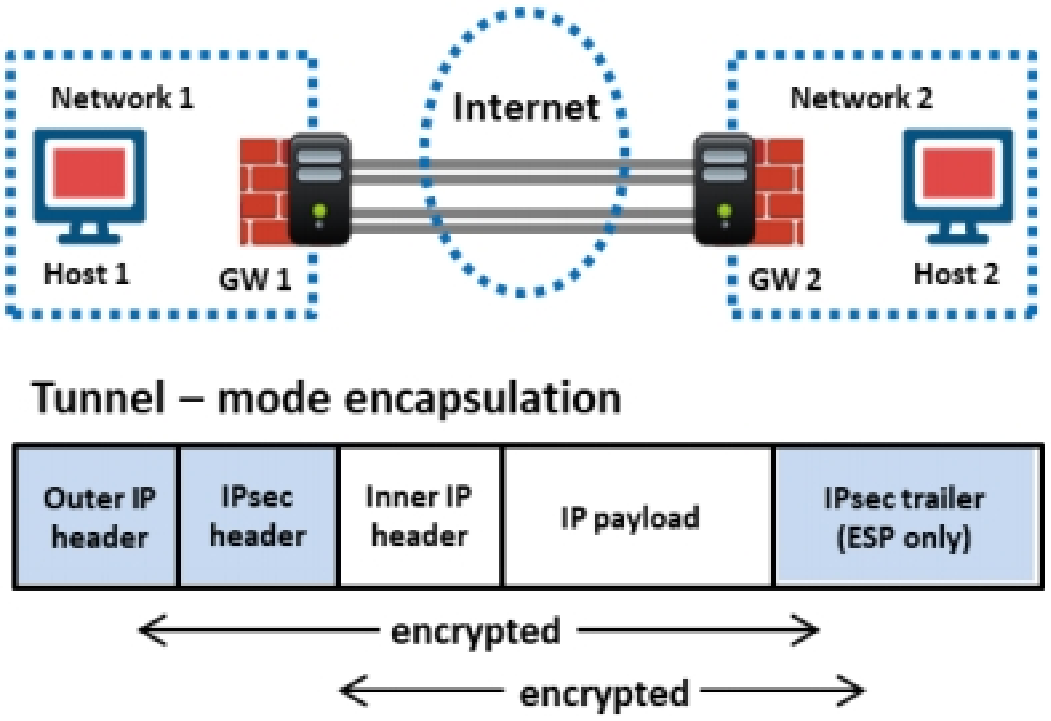


Figure 11.37: Tunnel mode encapsulation

IPsec Architecture

IPsec offers security services at the network layer. This provides the freedom to select the required security protocols as well as the algorithms used for services. To provide the requested services, the corresponding cryptographic keys can be employed, if required. Security services offered by IPsec include access control, data origin authentication, connectionless integrity, anti-replay, and confidentiality. To meet these objectives, IPsec uses two traffic security protocols, AH and ESP, as well as cryptographic key management protocols and procedures.

The protocol structure of the IPsec architecture is as follows.

- **Authentication Header (AH)**: It offers integrity and data origin authentication, with optional anti-replay features.
- **Encapsulating Security Payload (ESP)**: It offers all the services offered by AH as well as confidentiality.
- **IPsec Domain of Interpretation (DOI)**: It defines the payload formats, types of exchange, and naming conventions for security information such as cryptographic algorithms or security policies. IPsec DOI instantiates ISAKMP for use with IP when IP uses ISAKMP to negotiate security associations.
- **Internet Security Association and Key Management Protocol (ISAKMP)**: It is a key protocol in the IPsec architecture that establishes the required security for various communications over the Internet, such as government, private, and commercial

communications, by combining the security concepts of authentication, key management, and security associations.

- **Policy**: IPsec policies are useful in providing network security. They define when and how to secure data, as well as security methods to use at different levels in the network. One can configure IPsec policies to meet the security requirements of a system, domain, site, organizational unit, and so on.

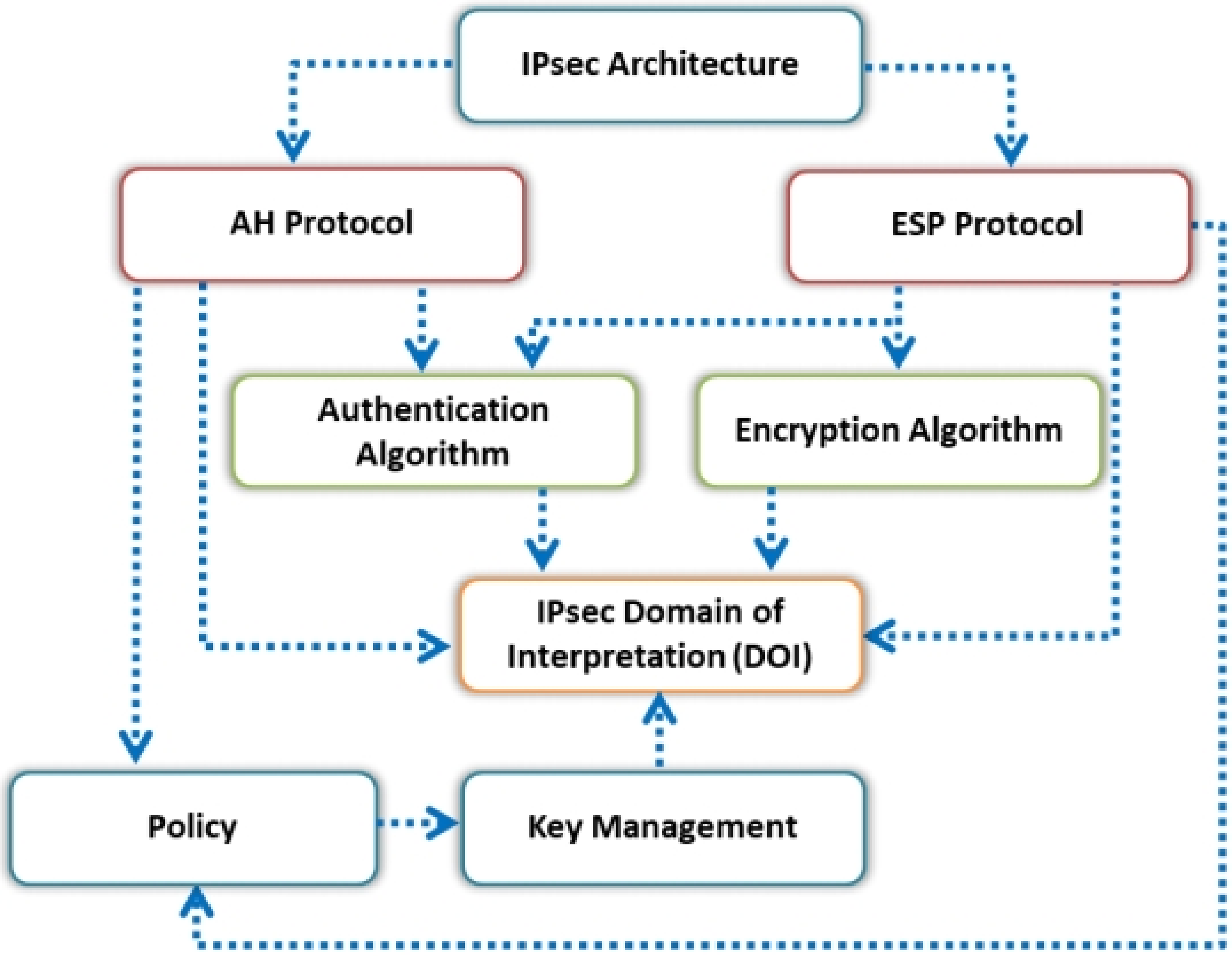
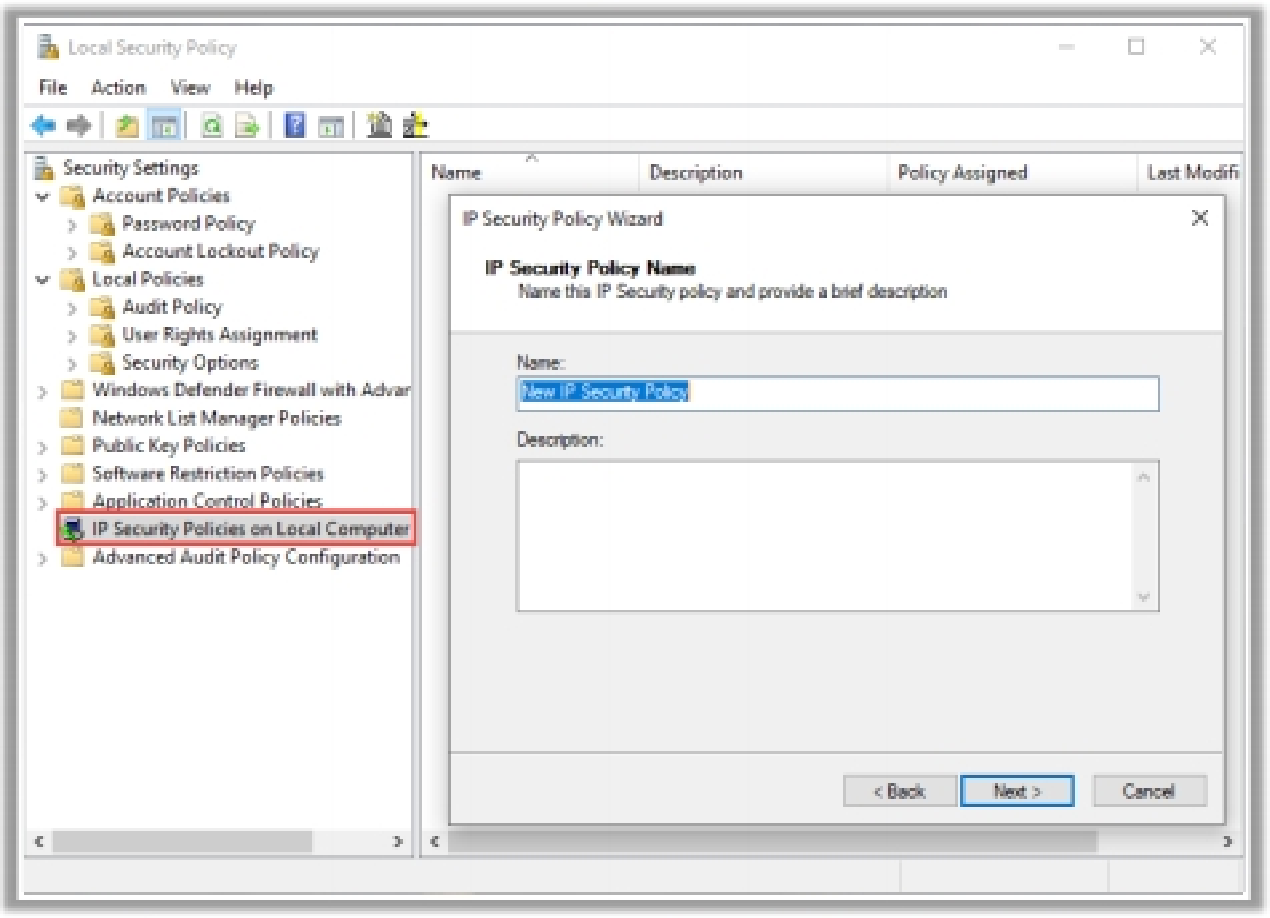

IPsec Architecture
AH Protocol
ESP Protocol
Authentication Algorithm
Encryption Algorithm
IPsec Domain of Interpretation (DOI)
Policy
Key Management

Figure 11.38: IPsec architecture

IPsec Authentication and Confidentiality

- IPsec uses two different security services for authentication and confidentiality
  - **Authentication Header (AH)**: Provides the data authentication of the sender
  - **Encapsulation Security Payload (ESP)**: Provides both the data authentication and encryption (confidentiality) of the sender


Local Security Policy
File
Action
View
Help
Security Settings
Account Policies
Password Policy
Account Lockout Policy
Local Policies
Audit Policy
User Rights Assignment
Security Options
Windows Defender Firewall with Advar
Network List Manager Policies
Public Key Policies
Software Restriction Policies
Application Control Policies
IP Security Policies on Local Computer
Advanced Audit Policy Configuration
Name
Description
Policy Assigned
Last Modifi
< Back
Next >
Cancel

IP Security Policy Wizard

IP Security Policy Name

Name this IP Security policy and provide a brief description

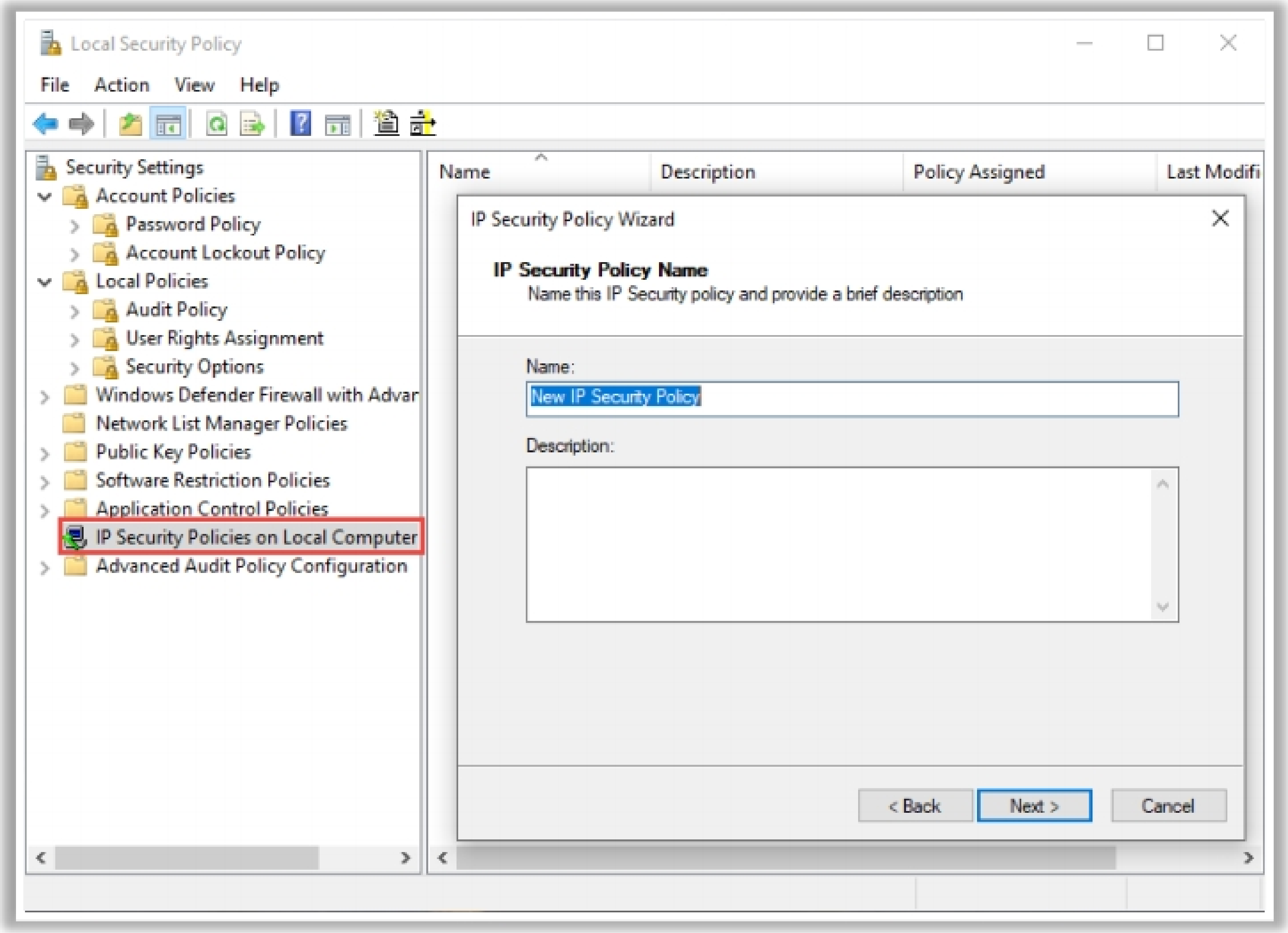
Name: New IP Security Policy

Description:

IPsec Authentication and Confidentiality

IPsec uses two different security services for authentication and confidentiality.

- **Authentication Header (AH)**: It is useful in providing connectionless integrity and data origin authentication for IP datagrams and anti-replay protection for the data payload and some portions of the IP header of each packet. However, it does not support data confidentiality (no encryption). A receiver can select the service to protect against replays, which is an optional service on establishing a security association (SA).
- **Encapsulation Security Payload (ESP)**: In addition to the services (data origin authentication, connectionless integrity, and anti-replay service) provided by AH, the ESP protocol offers confidentiality. Unlike AH, ESP does not provide integrity and authentication for the entire IP packet in the transport mode. ESP can be applied alone, in conjunction with AH, or in a nested manner. It protects only the IP data payload in the default setting. In the tunnel mode, it protects both the payload and IP header.


Local Security Policy
File
Action
View
Help
Security Settings
Account Policies
Password Policy
Account Lockout Policy
Local Policies
Audit Policy
User Rights Assignment
Security Options
Windows Defender Firewall with Advar
Network List Manager Policies
Public Key Policies
Software Restriction Policies
Application Control Policies
IP Security Policies on Local Computer
Advanced Audit Policy Configuration
Name
Description
Policy Assigned
Last Modifi
< Back
Next >
Cancel

IP Security Policy Wizard

IP Security Policy Name
Name this IP Security policy and provide a brief description

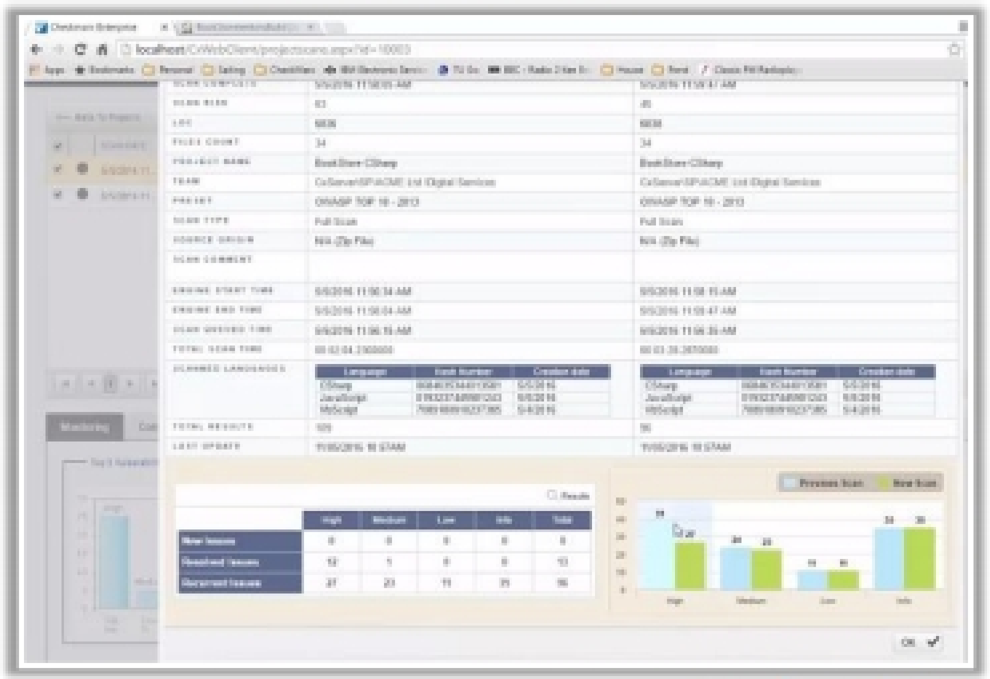
Name: New IP Security Policy

Description:

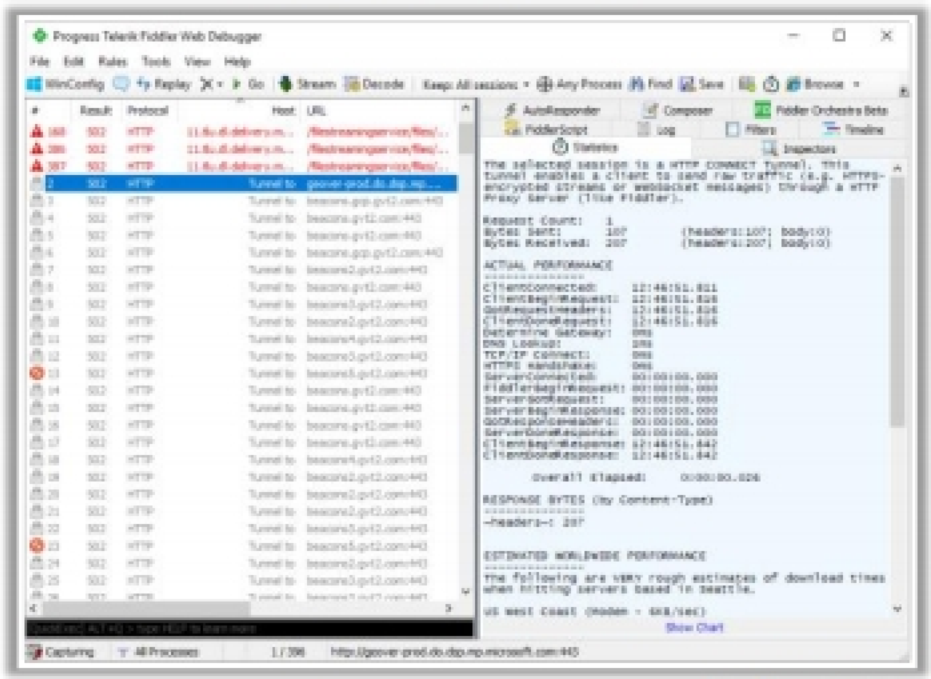
Figure 11.39: Screenshot of local IPsec policy on Windows

Session Hijacking Prevention Tools

CxSAST	CxSAST is a unique source code analysis solution that provides tools to identify, track, and repair technical and logical flaws in the source code
Fiddler	Fiddler is used for the security testing of web applications, such as decrypting HTTPS traffic and manipulating requests using a MITM decryption technique



https://www.checkmarx.com



https://www.telerik.com

Session Hijacking Prevention Tools:
- Nessus (*https://www.tenable.com*)
- Netsparker (*https://www.netsparker.com*)

Session Hijacking Prevention Tools

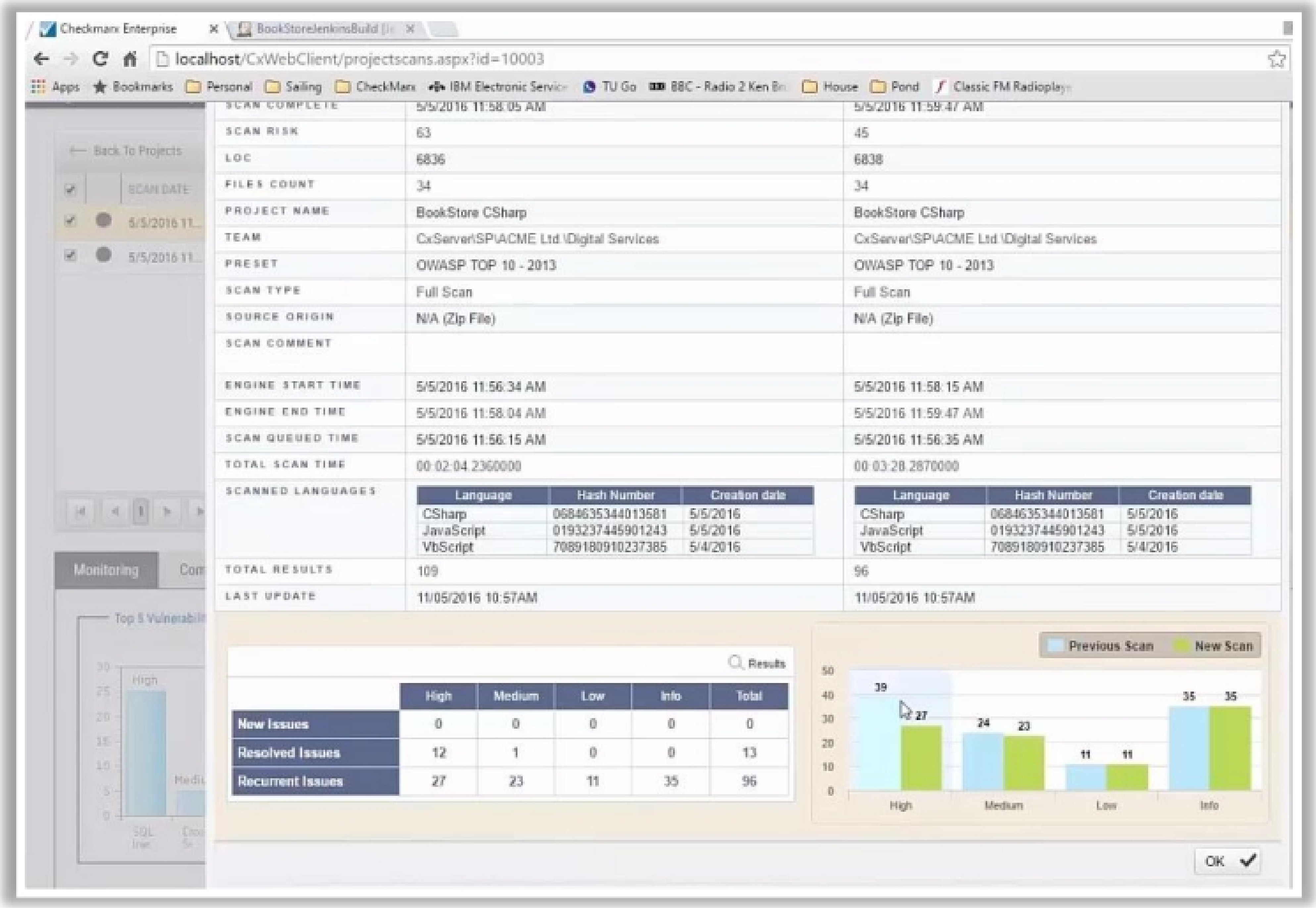
To prevent session hijacking, the security testing of web applications and the analysis of static code to identify vulnerabilities in web applications are required. Identifying vulnerabilities at an early stage helps in implementing security measures to protect against session hijacking attacks.

- **CxSAST**

  Source: *https://www.checkmarx.com*

  Checkmarx CxSAST is a unique source-code analysis solution that provides tools for identifying, tracking, and repairing technical and logical flaws in source code, such as security vulnerabilities, compliance issues, and business logic problems. CxSAST supports open-source analysis (CxOSA), enabling licensing and compliance management, vulnerability alerts, policy enforcement, and reporting. This tool supports a wide range of OS platforms, programming languages, and frameworks.

  Security professionals can use this tool to prevent various session hijacking attacks such as MITM attacks, session fixation attacks, and XSS attacks.


Checkmarx Enterprise
BookStoreJenkinsBuild
localhost/CxWebClient/projectscans.aspx?id=10003
Apps
Bookmarks
Personal
Sailing
CheckMarx
IBM Electronic Service
TU Go
BBC - Radio 2 Ken Br
House
Pond
Classic FM Radioplayer
Back To Projects
SCAN DATE
5/5/2016 11...
5/5/2016 11...
Monitoring
Com
Top 5 Vulnerabili
High
Medi
SQL Inj
Previous Scan
New Scan
50
40
30
20
10
0
39
27
24
23
11
11
35
35
High
Medium
Low
Info
Results
OK

SCAN COMPLETE	5/5/2016 11:58:05 AM	5/5/2016 11:59:47 AM
SCAN RISK	63	45
LOC	6836	6838
FILES COUNT	34	34
PROJECT NAME	BookStore CSharp	BookStore CSharp
TEAM	CxServer\SP\ACME Ltd \Digital Services	CxServer\SP\ACME Ltd \Digital Services
PRESET	OWASP TOP 10 - 2013	OWASP TOP 10 - 2013
SCAN TYPE	Full Scan	Full Scan
SOURCE ORIGIN	N/A (Zip File)	N/A (Zip File)
SCAN COMMENT		
ENGINE START TIME	5/5/2016 11:56:34 AM	5/5/2016 11:58:15 AM
ENGINE END TIME	5/5/2016 11:58:04 AM	5/5/2016 11:59:47 AM
SCAN QUEUED TIME	5/5/2016 11:56:15 AM	5/5/2016 11:56:35 AM
TOTAL SCAN TIME	00:02:04.2360000	00:03:28.2870000
SCANNED LANGUAGES	Language / Hash Number / Creation date CSharp / 0684635344013581 / 5/5/2016 JavaScript / 0193237445901243 / 5/5/2016 VbScript / 7089180910237385 / 5/4/2016	Language / Hash Number / Creation date CSharp / 0684635344013581 / 5/5/2016 JavaScript / 0193237445901243 / 5/5/2016 VbScript / 7089180910237385 / 5/4/2016
TOTAL RESULTS	109	96
LAST UPDATE	11/05/2016 10:57AM	11/05/2016 10:57AM

	High	Medium	Low	Info	Total
New Issues	0	0	0	0	0
Resolved Issues	12	1	0	0	13
Recurrent Issues	27	23	11	35	96

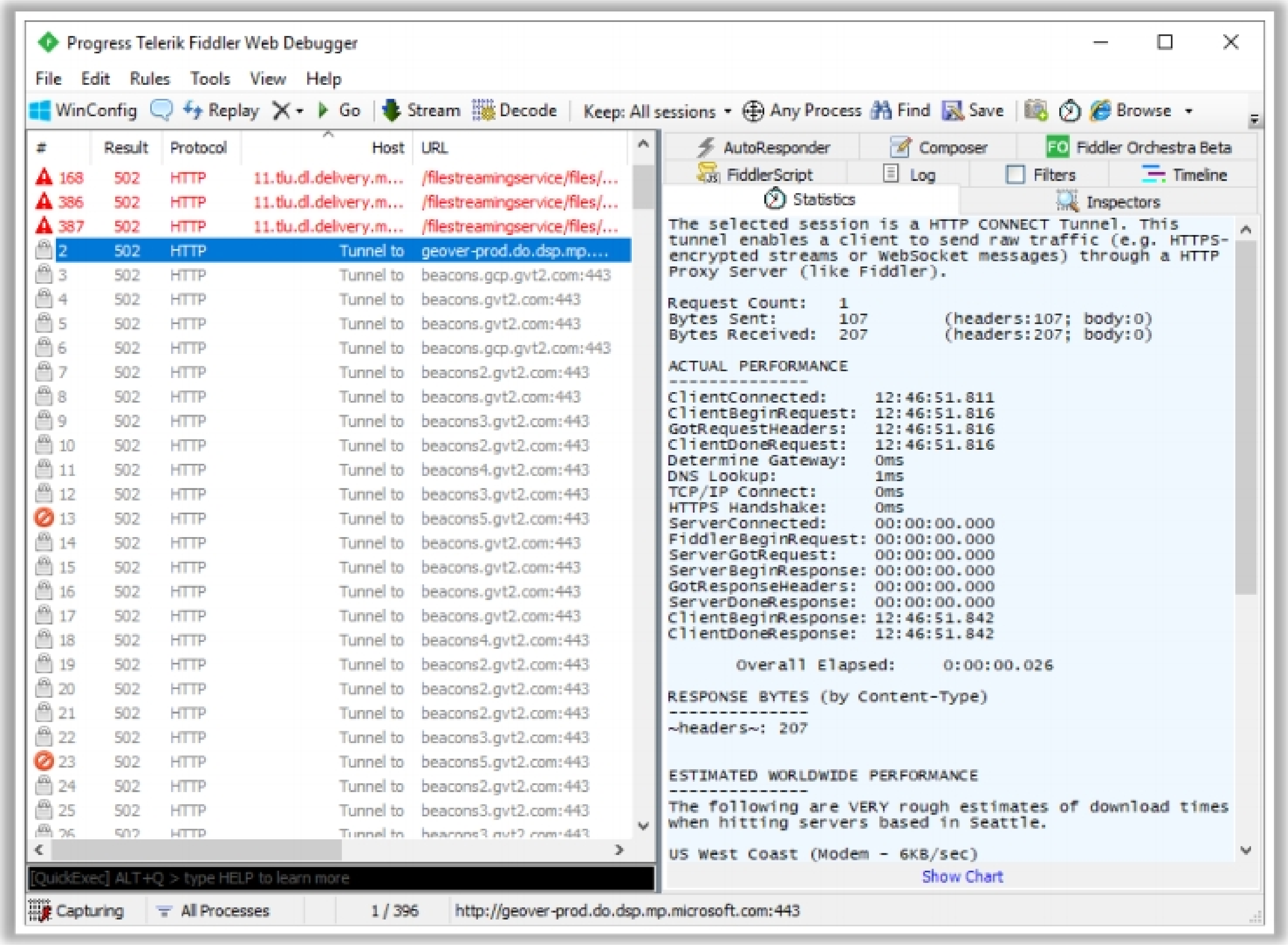
Figure 11.40: Screenshot of CxSAST

- **Fiddler**

  Source: *https://www.telerik.com*

  Fiddler is used for performing web-application security tests such as the decryption of HTTPS traffic and manipulation of requests using an MITM decryption technique. Fiddler is a web debugging proxy that logs all HTTP(S) traffic between a computer and the Internet.

  Security professionals can use Fiddler to test web applications by debugging the traffic from systems as well as manipulating and editing web sessions.


Progress Telerik Fiddler Web Debugger
File Edit Rules Tools View Help
WinConfig
Replay
Go
Stream
Decode
Keep: All sessions
Any Process
Find
Save
Browse
AutoResponder
Composer
FO Fiddler Orchestra Beta
FiddlerScript
Log
Filters
Timeline
Statistics
Inspectors
Show Chart
[QuickExec] ALT+Q > type HELP to learn more
Capturing
All Processes
1 / 396
http://geover-prod.do.dsp.mp.microsoft.com:443

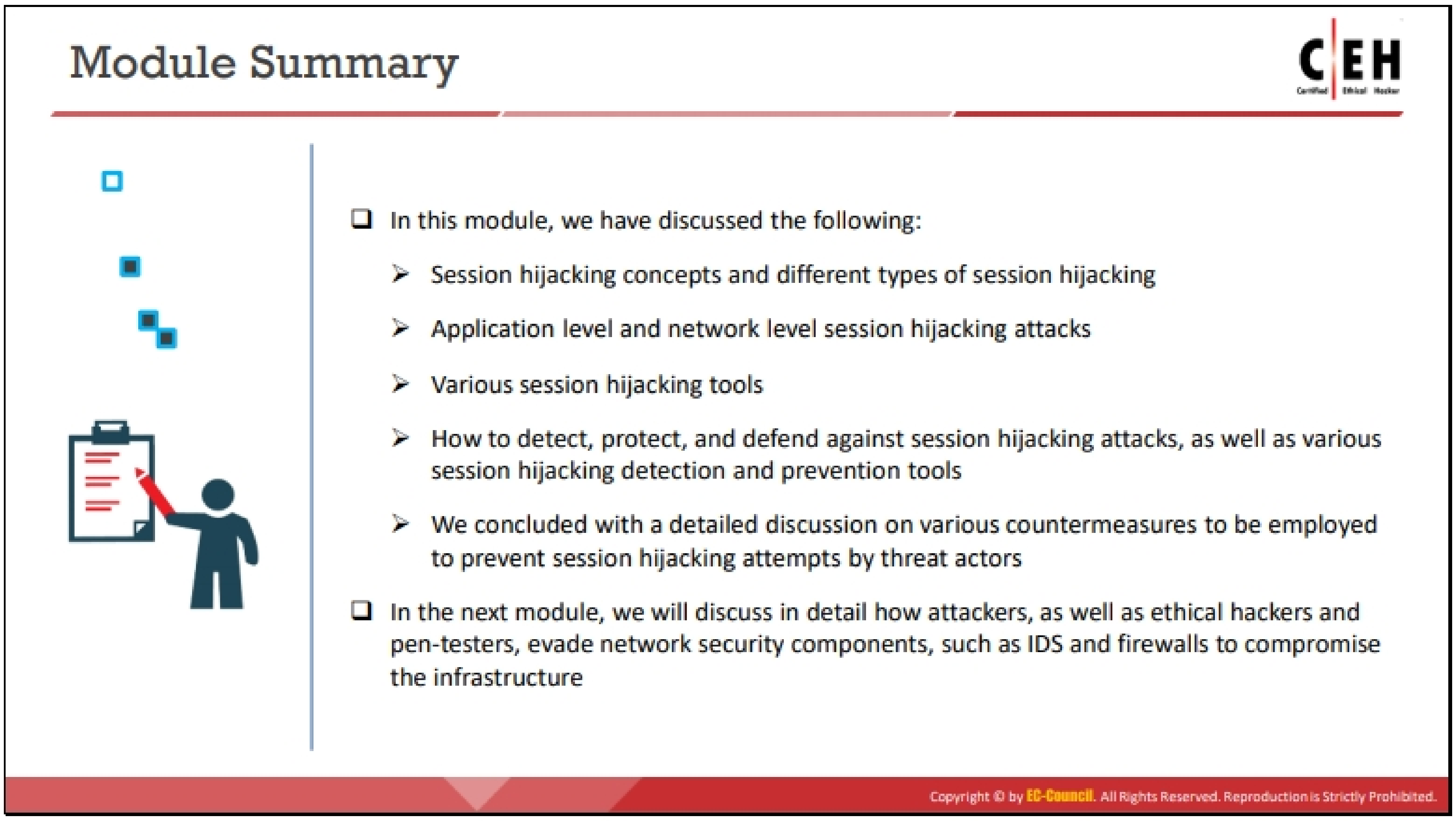
#	Result	Protocol	Host	URL
168	502	HTTP	11.tlu.dl.delivery.m...	/filestreamingservice/files/...
386	502	HTTP	11.tlu.dl.delivery.m...	/filestreamingservice/files/...
387	502	HTTP	11.tlu.dl.delivery.m...	/filestreamingservice/files/...
2	502	HTTP	Tunnel to	geover-prod.do.dsp.mp....
3	502	HTTP	Tunnel to	beacons.gcp.gvt2.com:443
4	502	HTTP	Tunnel to	beacons.gvt2.com:443
5	502	HTTP	Tunnel to	beacons.gvt2.com:443
6	502	HTTP	Tunnel to	beacons.gcp.gvt2.com:443
7	502	HTTP	Tunnel to	beacons2.gvt2.com:443
8	502	HTTP	Tunnel to	beacons.gvt2.com:443
9	502	HTTP	Tunnel to	beacons3.gvt2.com:443
10	502	HTTP	Tunnel to	beacons2.gvt2.com:443
11	502	HTTP	Tunnel to	beacons4.gvt2.com:443
12	502	HTTP	Tunnel to	beacons3.gvt2.com:443
13	502	HTTP	Tunnel to	beacons5.gvt2.com:443
14	502	HTTP	Tunnel to	beacons.gvt2.com:443
15	502	HTTP	Tunnel to	beacons.gvt2.com:443
16	502	HTTP	Tunnel to	beacons.gvt2.com:443
17	502	HTTP	Tunnel to	beacons.gvt2.com:443
18	502	HTTP	Tunnel to	beacons4.gvt2.com:443
19	502	HTTP	Tunnel to	beacons2.gvt2.com:443
20	502	HTTP	Tunnel to	beacons2.gvt2.com:443
21	502	HTTP	Tunnel to	beacons2.gvt2.com:443
22	502	HTTP	Tunnel to	beacons3.gvt2.com:443
23	502	HTTP	Tunnel to	beacons5.gvt2.com:443
24	502	HTTP	Tunnel to	beacons2.gvt2.com:443
25	502	HTTP	Tunnel to	beacons3.gvt2.com:443
26	502	HTTP	Tunnel to	beacons3.gvt2.com:443

```
The selected session is a HTTP CONNECT Tunnel. This
tunnel enables a client to send raw traffic (e.g. HTTPS-
encrypted streams or WebSocket messages) through a HTTP
Proxy Server (like Fiddler).

Request Count:   1
Bytes Sent:      107        (headers:107; body:0)
Bytes Received:  207        (headers:207; body:0)

ACTUAL PERFORMANCE
--------------
ClientConnected:     12:46:51.811
ClientBeginRequest:  12:46:51.816
GotRequestHeaders:   12:46:51.816
ClientDoneRequest:   12:46:51.816
Determine Gateway:   0ms
DNS Lookup:          1ms
TCP/IP Connect:      0ms
HTTPS Handshake:     0ms
ServerConnected:     00:00:00.000
FiddlerBeginRequest: 00:00:00.000
ServerGotRequest:    00:00:00.000
ServerBeginResponse: 00:00:00.000
GotResponseHeaders:  00:00:00.000
ServerDoneResponse:  00:00:00.000
ClientBeginResponse: 12:46:51.842
ClientDoneResponse:  12:46:51.842

       Overall Elapsed:     0:00:00.026

RESPONSE BYTES (by Content-Type)
--------------
~headers~: 207


ESTIMATED WORLDWIDE PERFORMANCE
--------------
The following are VERY rough estimates of download times
when hitting servers based in Seattle.

US West Coast (Modem - 6KB/sec)
```

Figure 11.41: Screenshot of Fiddler

The following are some additional session hijacking prevention tools:

- Nessus (*https://www.tenable.com*)
- Netsparker (*https://www.netsparker.com*)
- Wapiti (*http://wapiti.sourceforge.net*)
- WebWatchBot (*https://www.exclamationsoft.com*)



Module Summary

- ☐ In this module, we have discussed the following:
  - ➢ Session hijacking concepts and different types of session hijacking
  - ➢ Application level and network level session hijacking attacks
  - ➢ Various session hijacking tools
  - ➢ How to detect, protect, and defend against session hijacking attacks, as well as various session hijacking detection and prevention tools
  - ➢ We concluded with a detailed discussion on various countermeasures to be employed to prevent session hijacking attempts by threat actors
- ☐ In the next module, we will discuss in detail how attackers, as well as ethical hackers and pen-testers, evade network security components, such as IDS and firewalls to compromise the infrastructure

Module Summary

In this module, we discussed concepts related to session hijacking, along with different types of session hijacking. We also discussed in detail application level and network level session hijacking attacks. Furthermore, this module presented various session hijacking tools. It also discussed how to detect, protect, and defend against session hijacking attacks, in addition to various session hijacking detection and prevention tools. This module ended with a detailed discussion on various countermeasures to be employed to prevent session hijacking attempts by threat actors.

In the next module, we will discuss in detail how attackers, as well as ethical hackers and pen-testers, evade network security components such as IDSs and firewalls to compromise network infrastructure.